

A Model-Driven Approach for Developing Three-Dimensional User Interfaces of Information Systems in a Principle-based Way

By Juan Manuel González Calleros

A dissertation submitted in fulfilment of the requirements
for the degree of

Doctor of Philosophy in Economics and
Management Sciences
of the Université catholique de Louvain

Committee in charge:

Prof. Jean Vanderdonckt, Université catholique de Louvain, Advisor

Prof. Manuel Kolp, Université catholique de Louvain, Examiner

Ptof. Monique Noirhomme, Facultés Univ. Notre Dame de la Paix, Examiner

Prof. Karin Coninx, University of Hasselt, Reader

Prof. Jaime Muñoz Arteaga, Universidad Autonoma de Aguascalientes, Reader

Prof. Per Agrell, Université catholique de Louvain, President of Jury

4 February 2010

Acknowledgements

One brick after another a skyscraper is build. Similarly, step by step this thesis has been finished. Being the architect of this work I had nothing but to thanks all the actors that play a key role in constructing this thesis. Fortunately, during the whole journey I was accompanied by my family, friends and colleagues.

Siendo el pilar en mi vida y la base de lo que soy, mi familia nunca dejo de apoyarme pese a la distancia física que nos separaba. A mis padres, Rodolfo González Montealegre y Blanca Calleros Sosa, les puedo decir gracias por apoyar el sueño de aquel niño que insistía con estudiar un doctorado. El camino no fue fácil, ver partir a ese niño en busca de ese sueño y motivarlo pese a que eso implicara dejarlo de ver será algo que siempre agradeceré. Padres les doy las gracias por hacer de mí lo que soy y estar siempre apoyándose. Para mis hermanos, Rodolfo, Claudia y Verónica, siempre solidarios, gracias por ser mis mejores amigos, este logro se debe en mucho a su constante apoyo.

Josefina eres mi fortaleza, mi inspiración, la motivación que hace de mí una mejor persona. Estuviste ahí en los tiempos difíciles y disfrutaste conmigo las mieles del éxito. A Dios doy gracias por dejar que un día nuestros caminos coincidieran. Me has dado dos increíbles regalos, José Manuel y Andrea, que han redondeado nuestra felicidad. Gracias chica por guiarme hasta aquí, esto se logro con tu ayuda.

During my studies, mostly, I met Chileans and I learnt a lot about wine, barbecues, food and Chilean culture. They showed me all the splendour of their country and the nobility of their citizens. Chilean friends will be always in my heart and for sure we will meet again somewhere.

The BCHI members for their fruitful sharing through our research experiences including: meetings, invited talks, projects involvement. Particularly to Adrian Stanciulescu who used me constantly as a participant in his research experiments and immortalized my name in his PhD thesis through his examples.

To all my football colleagues for our usual Friday night match that help me to lose the stress of the week.

It is hard to find the words to express all the gratitude that I have to my advisor Jean Vanderdonckt. Since my arrival in Belgium he had nothing but a positive attitude, a constant encouragement, a constant care first on the human aspects then to the research activities. Thank you very much for letting me be part of the lab first, then to invite me in the different research activities in which I have been in-

volved. Thank you for sharing all your experience, anecdotes, and background you have conducting research. Thank you for those constant and fruitful meetings when you always have a lot of ideas to discuss.

A Jaime Muñoz por confiar en mí y mostrarme la puerta que tenía que tocar para hacer un doctorado en el extranjero. Por su constante apoyo y su guía. Por la colaboración en los diferentes proyectos.

Many thanks to Professors Karin Coninx, Monique Noirhomme, Manuel Kolp, and Jaime Muñoz Arteaga for accepting to participate to the jury of this thesis.

Special thanks to all members of the NexOF-RA European project (in particular the declarative language for user interface authoring and the Context Model and Universal API research teams) and to all members of the W3C Model Based User Interface Incubator group for their fruitful discussions.

This thesis was accomplished thanks to the support of:

- The Human project (Model-based Analysis of Human Errors during Aircraft Cockpit System Design), supported by the 7th Framework Programme of the European Commission, under contract FP7-AAT-2007-RTD-1 (www.human.aero).
- The ALBAN program, the high level European scholarship programme specifically addressed to Latin America for benefiting thus from the excellence of the Higher Education in the European Union, supported by the European Commission, under contract E04D033272MX (www.programalban.org).
- The CONACYT, the high level Mexican scholarship programme for benefiting thus from the excellence of the Higher Education, under contract 166002 (www.conacyt.mx).
- The Information Systems (ISYS) Unit of Louvain School of Management (LSM) of Université catholique de Louvain.
- The UsiXML Consortium, for User Interface eXtensible Markup Language (www.usixml.org).
- The SIMILAR network of excellence, the European research task force creating human-machine interfaces similar to human-human communication, supported by the 6th Framework Program of the European Commission, under contract FP6-IST1-2003-507609 (www.similar.cc).
- The ITEA2 Call 3 UsiXML Project under Contract #2008026.
- The PROMEP net Project under Contract UAA-CA-48.

Table of Contents

ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	5
TABLE OF FIGURES	9
TABLE OF TABLES	15
CHAPTER 1 INTRODUCTION	17
1.1 WHAT IS A 3D USER INTERFACE?	17
1.2 WHY 3D USER INTERFACES?	19
1.3 WHY NOT 3D USER INTERFACES?	20
1.4 CONCERNS	20
1.5 THESIS	22
1.5.1 Thesis statement	22
1.5.2 Focus of the thesis	23
1.5.3 Some Definitions	26
1.5.4 Target Audience	27
1.6 READING MAP	29
CHAPTER 2 STATE OF THE ART	33
2.1 SOFTWARE SUPPORT	33
2.2 METHODS FOR 3DUI DEVELOPMENT	36
2.2.1 Web & Information System Engineering Lab	37
2.2.2 CoGenIVE	38
2.2.3 Interaction Techniques Markup Language	40
2.2.4 Contigra	41
2.2.5 TRES-D Methodology	41
2.2.6 Desktop Based Methods	43
2.2.7 Participative Method	44
2.2.8 Task analysis for building VEs	44
2.2.9 Comparative Analysis on Model-based Developments	45
2.3 TRANSFORMATION SYSTEMS COMPARISON	46
2.4 USER INTERFACE DESCRIPTION LANGUAGES	49
2.5 3DUIs EVALUATION METHODS	51
2.6 CONCLUSION	52
2.6.1 Shortcomings Identified from the Literature Review	54

2.6.2	<i>Ontological Requirements</i> -----	57
2.6.3	<i>Methodological Requirements</i> -----	58
CHAPTER 3 ONTOLOGY OF THREE-DIMENSIONAL USER INTERFACES---		69
3.1	ONTOLOGY FOR 3DUI SPECIFICATION.....	70
3.2	TASK MODEL.....	71
3.3	DOMAIN MODEL.....	75
3.4	ABSTRACT USER INTERFACE MODEL	77
3.5	CONCRETE USER INTERFACE MODEL	83
3.5.1	<i>Three-D Rendering of 2DUI Model</i> -----	83
3.5.2	<i>Virtual Three-D Graphical User Interface Model</i> -----	83
3.5.3	<i>3D Graphical Individual Components</i> -----	92
3.5.4	<i>Modelling Behaviour</i> -----	93
3.5.5	<i>Modelling Appearance</i> -----	103
3.5.6	<i>Modelling Geometry</i> -----	104
3.5.7	<i>Modelling Haptic Interaction</i> -----	107
3.6	CONCLUSION	111
CHAPTER 4 A METHOD FOR DEVELOPING 3DUIS IN A PRINCIPLE-BASED WAY -----		113
4.1	SOFTWARE DEVELOPMENT LIFE CYCLE OF 3DUI FOR INFORMATION SYSTEMS ...	114
4.2	STEP 1: TASK AND DOMAIN MODELLING.....	116
4.2.1	<i>Three Dimensional Task Patterns</i> -----	116
4.2.2	<i>Guidelines for Task Modelling</i> -----	117
4.2.3	<i>Running example: The Interactive Table task and concepts modelling</i> -----	121
4.3	STEP 2: FROM TASK AND DOMAIN MODEL TO AUI MODEL.....	122
4.3.1	<i>Guidelines for Abstract User Interface Modelling</i> -----	122
4.3.2	<i>Running example: The Interactive Table Abstract User Interface Modelling</i>	123
4.4	STEP 3: FROM AUI TO CUI.....	123
4.4.1	<i>Guidelines for Concrete User Interface Modelling</i> -----	124
4.4.2	<i>Running example: The Interactive Table Concrete User Interface</i> -----	129
4.5	STEP 4: FROM CUI TO FUI.....	131
4.5.1	<i>Guidelines for Final User Interface Development</i> -----	131
4.5.2	<i>Running example: : The Interactive Table Final User Interface</i> -----	136
4.6	ADDING A NEW CONCEPT TO THE ONTOLOGY	136
4.7	CONCLUSION	138
CHAPTER 5 TOWARDS A UIDL FOR 3DUI -----		139
5.1	UIDL SELECTION	140
5.2	UIDL SEMANTICS.....	141

5.3	UIDL SYNTAX.....	142
5.4	UIDL STYLISTICS.....	144
5.5	CONCLUSION.....	144
CHAPTER 6 SOFTWARE FOR SUPPORTING THE METHODOLOGY-----		147
6.1	MODEL EDITORS.....	149
6.2	DESIGN CRITICS: GUIDELINES EVALUATION.....	150
6.3	IMPLEMENTATION TOOLS.....	153
6.4	TRANSFORMATION TOOL.....	155
6.5	HAPTIC BROWSER.....	157
6.6	CONCLUSION.....	158
CHAPTER 7 VALIDATION -----		161
7.1	CASE STUDIES.....	161
7.1.1	<i>Case Study 1: The virtual polling system-----</i>	<i>162</i>
7.1.2	<i>Case Study 2: The Student- Trainer Application -----</i>	<i>191</i>
7.1.3	<i>Case Study 3: Advanced Flight Management System handled by a 3D Navigation Display-----</i>	<i>197</i>
7.1.4	<i>Case Study 4: Haptic application -----</i>	<i>216</i>
7.2	INTERNAL VALIDATION.....	217
7.2.1	<i>Why should the User Interface development method be formal?-----</i>	<i>217</i>
7.2.2	<i>Why should the User Interface development method be standardized? -----</i>	<i>218</i>
7.2.3	<i>What do we need to have a formal User Interface development method to design the 3DUI?-----</i>	<i>219</i>
7.2.4	<i>Requirements evaluation -----</i>	<i>219</i>
7.3	EXTERNAL EVALUATION.....	222
7.4	CONCLUSION.....	226
CHAPTER 8 CONCLUSION -----		229
8.1	CONTENTS OF THIS DISSERTATION.....	229
8.2	SUMMARY OF CONTRIBUTIONS.....	230
8.3	ON THE PROMISES OF MDA.....	232
8.4	FUTURE WORK.....	233
REFERENCES -----		239
APPENDIX A. TRANSFORMATIONAL RULES -----		269
APPENDIX B. USIXML ONTOLOGY FOR USER INTERFACE SPECIFICATION		285
APPENDIX C. THREE-DIMENSIONAL WIDGETS REPRESENTATION-----		296
APPENDIX D. CANONICAL LIST OF TASK TYPES -----		324

APPENDIX E. THREE DIMENSIONAL USER INTERFACES TAXONOMY	----- 339
APPENDIX F. STATE OF THE ART	----- 355
APPENDIX G. TASK PATTERNS	----- 368

Table of Figures

FIGURE 1-1. TASKGALLERY, AN EXAMPLE OF A 2D TASK IN 3D.....	18
FIGURE 1-2. BENEFITS OF THE IDENTIFIED CONCERNS OF 3DUIS.....	22
FIGURE 1-3. TYPOLOGY OF ORGANISATIONAL BEHAVIOURS.....	25
FIGURE 1-4. SOME INFORMATION SYSTEMS OUT OF THE SCOPE OF THIS THESIS.....	27
FIGURE 2-1. SLIDER DESIGN USING VIVATY STUDIO.....	34
FIGURE 2-2. CAMELEON-BASED METHODOLOGY LAYERED PROFILE.....	36
FIGURE 2-3. VRXML OBJECT PROPERTY DIALOG.....	39
FIGURE 2-4. COGENIVE DEVELOPMENT PROCESS.....	39
FIGURE 2-5. TRES-D DEVELOPMENT PROCESS.....	42
FIGURE 2-6. TOOLS COMPARISON WITH RESPECT TO PERFORMANCE AND MAINTAINABILITY.....	48
FIGURE 3-1. GENERAL SCHEMA FOR ONTOLOGY DERIVATION.....	70
FIGURE 3-2. THE DISTRIBUTION OF MODELS ACCORDING TO THE MDA CLASSIFICATION.....	71
FIGURE 3-3. TASK META-MODEL.....	79
FIGURE 3-4. DOMAIN META-MODEL.....	80
FIGURE 3-5. ABSTRACT USER INTERFACE META-MODEL.....	81
FIGURE 3-6. EXTENSION OF THE 2D CONCRETE USER INTERFACE.....	82
FIGURE 3-7. 3DGCIO META-AGGREGATION RELATIONSHIPS.....	85
FIGURE 3-8. THREE-D RENDERING OF 2DUIS.....	86
FIGURE 3-9. 2D MENUS.....	86
FIGURE 3-10. DIGITAL VERSION OF THE MENU.....	87
FIGURE 3-11. SCENE WITH SEVERAL PLANES.....	87
FIGURE 3-12. WALL CONTAINER WITH TRANSPARENCY.....	88
FIGURE 3-13. REGULAR PRISMS.....	88
FIGURE 3-14. PYRAMID.....	88
FIGURE 3-15. SPHERE CONTAINER.....	88
FIGURE 3-16. EXAMPLES OF 3D SLIDERS.....	89
FIGURE 3-17. COMPUTING THE CURRENT VALUE ON A 3D SLIDER.....	89
FIGURE 3-18. HOW TO PAINT A 3D SLIDER.....	89
FIGURE 3-19. THREE-DIMENSIONAL SLIDERS TAXONOMY.....	90
FIGURE 3-20. EVENT CLASS FOR THE BEHAVIOUR MODEL.....	97
FIGURE 3-21. APPEARANCE META-MODEL.....	98
FIGURE 3-22. GEOMETRY META-MODEL.....	99
FIGURE 3-23. GROUPING META-MOD.....	100
FIGURE 3-24. THREE-DIMENSIONAL HAPTIC COMPONENTS.....	101
FIGURE 3-25. HAPTIC EXTENSION TO USIXML CONCRETE USER INTERFACE MODEL.....	102

FIGURE 3-26. TIMELINE OF CONTRIBUTIONS TO UsiXML MODELS.....	111
FIGURE 4-1. GENERAL SCHEMA FOR METHOD DERIVATION.	113
FIGURE 4-2. OUTLINE OF THE METHOD FOR DEVELOPING 3DUIS.....	115
FIGURE 4-3. AMBIGUITY AVOIDANCE ON TASK MODELLING.	118
FIGURE 4-4. GUIDELINE FOR TASK NAMING.	118
FIGURE 4-5. GUIDELINE FOR TASK NAMING.	119
FIGURE 4-6. OVERVIEW OF THE AIO SELECTION.	123
FIGURE 4-7. VIRTUAL OFFICE TASK MODEL.	125
FIGURE 4-8. ADAPTING AN AUI THAT FURTHER WILL NEED NAVIGATION FACETS.....	126
FIGURE 4-9. VIRTUAL OFFICE CONCRETE MODEL.....	127
FIGURE 4-10. GRAPHICAL REPRESENTATION FOR A TOGGLE BUTTON.	128
FIGURE 4-11. MOCK-UP OF THE CONTROL SCREEN UI. FIGURE 4-12. MOCK-UP OF THE INTERACTING TABLE.	130
FIGURE 4-13. MOCK-UP OF THE CONTROL SCREEN UI.	131
FIGURE 4-14. GUIDANCE IN VIRTUAL REALITY FOR REQUESTING NAVIGATION HELP.....	133
FIGURE 4-15. VIRTUAL OFFICE WITH AN INTERACTIVE TABLE.	134
FIGURE 4-16. TOP VIEW OF THE VIRTUAL TABLE RENDERED IN VRML.....	134
FIGURE 4-17. BIG SCREEN RENDERED IN VRML.....	135
FIGURE 4-18 IMPLEMENTATION DIAGRAM OF THE TOGGLE BUTTON.....	137
FIGURE 5-1. GENERAL SCHEMA FOR UIDL DERIVATION.	139
FIGURE 5-2. GENERATION OF UsiXML SPECIFICATION.	142
FIGURE 5-3. CORRESPONDENCE OF A CLASS IN UsiXML SPECIFICATION.	143
FIGURE 5-4. CORRESPONDENCE OF A RELATIONSHIP CLASS IN UsiXML SPECIFICATION.	143
FIGURE 5-5. CORRESPONDENCE OF THE INHERITANCE RELATIONSHIP.	143
FIGURE 6-1. GENERAL SCHEMA FOR UIDL DERIVATION.	147
FIGURE 6-2. SOFTWARE CHAIN FOR SUPPORTING THE METHODOLOGY.....	149
FIGURE 6-3. TASK, DOMAIN, ABSTRACT AND MAPPING MODELS SPECIFICATION.	150
FIGURE 6-4. ADAPTATION BASED ON ERGONOMIC RULE.....	150
FIGURE 6-5. BAD COLOUR COMBINATION.	151
FIGURE 6-6. USABILITY ADVISOR INTERFACE.	151
FIGURE 6-7. TDCIOS TOOLKIT IN VIVATY STUDIO.....	152
FIGURE 6-8. ALICE ENVIRONMENT.	152
FIGURE 6-9. GRAPH TRANSFORMATIONS AND TRANSFORMATIONAL DEVELOPMENT.....	156
FIGURE 6-10. FROM HTML TO A 3D SCENE.	157
FIGURE 6-11. STEPS OF THE IMPLEMENTATION OF THE TEMPLATE DB.	158
FIGURE 7-1. GENERAL SCHEME OF THE VALIDATION.....	161
FIGURE 7-2 MAPPING BETWEEN DOMAIN CONCEPTS AND TASK MODEL.....	163
FIGURE 7-3. CONSOLIDATED TASK MODEL OF THE POLLING SYSTEM.....	165
FIGURE 7-4. POSSIBLE DEVELOPMENT PATHS.	165

FIGURE 7-5. IDEALXML MAPPING FROM T&D MODEL TO ABSTRACT MODEL A.....	170
FIGURE 7-6. LHS RULE FOR CREATING NAVIGATION FACET FOR ABSTRACT CONTAINERS.....	172
FIGURE 7-7. RHS RULE FOR CREATING NAVIGATION FACET FOR ABSTRACT CONTAINERS.	172
FIGURE 7-8. NAC FOR CREATING NAVIGATION FACET FOR ABSTRACT CONTAINERS.	172
FIGURE 7-9. LHS RULE FOR CREATING NAVIGATION FACET FOR THE FIRST AC.....	173
FIGURE 7-10. NACS FOR CREATING NAVIGATION FACET FOR THE FIRST AC.	173
FIGURE 7-11. RHS RULE FOR CREATING NAVIGATION FACET FOR THE FIRST AC.	173
FIGURE 7-12. LHS RULE FOR CREATING NAVIGATION FACET FOR THE LAST AC.	174
FIGURE 7-13. RHS RULE FOR CREATING NAVIGATION FACET FOR THE LAST AC.....	174
FIGURE 7-14. NAC FOR CREATING NAVIGATION FACET FOR THE LAST AC.....	174
FIGURE 7-15. IDEALXML MAPPING FROM T&D MODEL TO ABSTRACT MODEL B.....	175
FIGURE 7-16. MOCK-UP OF THE UI.	184
FIGURE 7-17. POLLING SYSTEM RENDERED IN VRML.	189
FIGURE 7-18. HEXAHEDRON POLLING SYSTEM.	189
FIGURE 7-19. EDITION OF THE 3DUI IN MAYA.	190
FIGURE 7-20. THREE-D RENDERING OF THE 2D INTERFACE FOR THE POLLING SYSTEM.	190
FIGURE 7-21. TASK & CONCEPTS OF THE STUDENT/TRAINER SYSTEM.	191
FIGURE 7-22. RULE FOR CONSOLIDATION OF THE TASK MODEL.....	192
FIGURE 7-23. CONSOLIDATED TASK MODEL OF THE STUDENT/TRAINER SYSTEM.....	192
FIGURE 7-24. RULE FOR ABSTRACT CONTAINER ASSIGNATION FOR TASKS.	193
FIGURE 7-25. EACH LEAF TASK IS EXECUTED IN AN ABSTRACT INDIVIDUAL COMPONENT.	194
FIGURE 7-26. ABSTRACT USER INTERFACE MODEL.....	194
FIGURE 7-27. MAPPING RULE FOR SLIDER SELECTION.....	195
FIGURE 7-28. MAPPING RULE FOR OUTPUT TEXT SELECTION.	196
FIGURE 7-29. A FINAL RENDERING OF THE OF THE STUDENT/TRAINER SYSTEM.	197
FIGURE 7-30. TOOL CHAIN OF THE SYMBOLIC AHMI.	199
FIGURE 7-31. SYMBOLIC AHMI ARCHITECTURE.....	199
FIGURE 7-32. EVOLUTION OF THE SYMBOLIC AHMI ARCHITECTURE.	200
FIGURE 7-33. TASK MODEL OF THE CREATE TRAJECTORY TASK.....	204
FIGURE 7-34. ABSTRACT USER INTERFACE OF THE SELECT TRAJECTORY TASK.	205
FIGURE 7-35. SPECIAL GRAPHICAL INDIVIDUAL COMPONENTS USED IN THE AHMI.	206
FIGURE 7-36. LAYOUT OF THE AHMI NAVIGATION DISPLAY.....	207
FIGURE 7-37. SEMANTICS OF THE CUI MODEL OF THE NAVIGATION DISPLAY LAYOUT.....	208
FIGURE 7-38. MOCK-UP OF THE NAVIGATION DISPLAY LAYOUT.	208
FIGURE 7-39. AHMI NEGOTIATING EVOLUTION.	211
FIGURE 7-40. NAVIGATION DISPLAY RENDERED AS A 3DUI.....	212
FIGURE 7-41. TEST CASE: WWW.GREECE.COM.	213
FIGURE 7-42. CASE STUDY: WWW.OGRE3D.ORG.	214
FIGURE 7-43. OPTIONS.....	214

FIGURE 7-44. LARGE HAPGETS.....	215
FIGURE 7-45. CHOOSE HAPGETS.	215
FIGURE 7-46. PERSONAL SUBJECTIVE REQUIREMENTS EVALUATION.....	220
FIGURE 7-47. IBM CSUQ METRICS.	226
FIGURE A-1 CREATION OF A NODE WITH ATTRIBUTES	269
FIGURE A-2 NODE MODIFICATION (IDENTIFIED INSTANCE).....	269
FIGURE A-3 NODE MODIFICATION UNIDENTIFIED INSTANCE.....	270
FIGURE A-4 NEGATIVE APPLICATION CONDITION	270
FIGURE A-5 NEGATIVE APPLICATION CONDITION (2).....	271
FIGURE A-6 RULE WITH VARIABLE AND POSITIVE APPLICATION CONDITION	271
FIGURE A-7 TRANSFER OF AN ATTRIBUTE VALUE.....	272
FIGURE A-8 EDGE CREATION.....	272
FIGURE A-9 NODE DELETION.....	272
FIGURE B-1 THE CAMELEON REFERENCE FRAMEWORK FOR MULTI-TARGET UIs	285
FIGURE C-1 THREE-D TOGGLE BUTTON EVALUATION DIAGRAM	298
FIGURE C-2 THREE-DIMENSIONAL TOGGLE BUTTON TAXONOMY	299
FIGURE C-3 THREE-D OUTPUT TEXT EVALUATION DIAGRAM	301
FIGURE C-4 THREE-D INPUT TEXT EVALUATION DIAGRAM.....	302
FIGURE C-5 THREE-DIMENSIONAL TEXT COMPONENT TAXONOMY.....	303
FIGURE C-6 THREE-D COLOUR PICKER EVALUATION DIAGRAM.....	304
FIGURE C-7 THREE-DIMENSIONAL COLOUR PICKER	305
FIGURE C-8 THREE-D RADIO BUTTON EVALUATION DIAGRAM	306
FIGURE C-9 THREE-DIMENSIONAL RADIO BUTTON TAXONOMY.....	307
FIGURE C-10 THREE-D CHECK BOX EVALUATION DIAGRAM	308
FIGURE C-11 THREE-DIMENSIONAL CHECK BOX TAXONOMY	309
FIGURE C-12 THREE-D COMBO BOX EVALUATION DIAGRAM.....	310
FIGURE C-13 THREE-DIMENSIONAL COMBOBOX TAXONOMY	311
FIGURE C-14 THREE-DIMENSIONAL LISTBOX TAXONOMY	313
FIGURE C-15 THREE-D BUTTON EVALUATION DIAGRAM.....	315
FIGURE C-16 THREE-DIMENSIONAL BUTTON TAXONOMY	316
FIGURE C-17 THREE-D HYPERLINK EVALUATION DIAGRAM	318
FIGURE C-18 THREE-D MENU EVALUATION DIAGRAM.....	319
FIGURE C-19 THREE-D IMAGE COMPONENT EVALUATION DIAGRAM	321
FIGURE C-20 THREE-D VIDEO COMPONENT.....	323
FIGURE E-1 TAXONOMY OF SELECTION/MANIPULATION TECHNIQUES	340
FIGURE E-2 ADAPTATION TO THE TAXONOMY OF VIRTUAL METAPHORS	341
FIGURE E-3 CONTIGRA WIDGETS TAXONOMY.....	342
FIGURE E-4 SIMPLIFIED REPRESENTATION OF A "VIRTUALITY CONTINUUM"	343
FIGURE E-5 EXTENDED REPRESENTATION OF A "VIRTUALITY CONTINUUM"	344

FIGURE E-6 AN EXTENDED CONTINUUM OF USER INTERFACES	345
FIGURE E-7 THREE-DIMENSIONAL CAROUSELS IN DAILY LIFE AND COMPUTER INTERFACES ...	347
FIGURE E-8 REKIMOTO'S NAVICAM SYSTEM AND AUGMENTED INTERACTION 1995.....	347
FIGURE E-9 REKIMOTO'S AUGMENTED SURFACES	347
FIGURE E-10 KIYOKAWA ET AL. 2000	350
FIGURE E-11 MAGIC MIRROR.....	350
FIGURE E-12 IMAGE PLANE	350
FIGURE E-13 COLLAPSIBLE CYLINDRICAL TREES.....	350
FIGURE E-14 3DNA DESKTOP APPLICATION	351
FIGURE E-15 THE VIRTUAL LAPTOP WITH THE RENDERED IN VRML97.....	352
FIGURE E-16 VISUAL STUDIO SOFTWARE TOOL TO DEVELOP 2D GUI APPLICATIONS.....	353
FIGURE F-1 GAME USER INTERFACE CREATED WITH CRAZY EDDIES' TOOLKIT	356
FIGURE F-2 ANARK STUDIO ADDING BEHAVIOUR TO A BUTTON	356
FIGURE F-3 PUEBLA'S CATHEDRAL DESIGNED IN GOOGLE SKETCHUP	360
FIGURE F-4 VIVATY STUDIO.....	360
FIGURE F-5 SEAMLESS3D AN AVATAR STUDIO	361
FIGURE F-6 OGRE GAME RENDERING	361
FIGURE F-7 SPHEREXP THREE DIMENSIONAL DESKTOP	363
FIGURE F-8 DESKTOP IN 3DNA	363
FIGURE F-9 LOOKING GLASS DESKTOP	365
FIGURE F-10 CLARA WEB BROWSER.....	365
FIGURE F-11 SPACE TIME BROWSER	366
FIGURE F-12 EXIT REALITY	366
FIGURE G-1 NAVIGATION PATTERN.....	368
FIGURE G-2 TRAVEL PATTERN	369
FIGURE G-3 WAYFINDING PATTERN.....	370
FIGURE G-4 SELECT PATTERN	371
FIGURE G-5 MANIPULATION PATTERN	373
FIGURE G-6. SYSTEM CONTROL PATTERN.....	373

Table of tables

TABLE 1-1. NATURE OF THE TASK AND ITS FRONT-END.	18
TABLE 1-2. EXPRESSION OF INTERACTION STYLES IN TERMS OF TASK PARAMETERS [VAND00].	31
TABLE 1-3. EXPRESSION OF INTERACTION STYLES IN TERMS OF USER PARAMETERS [VAND00].	32
TABLE 2-1. TOOLKITS COMPARISON.	60
TABLE 2-2. MODEL-BASED METHODOLOGIES COMPARISON.	61
TABLE 2-3. COMPARISON OF TRANSFORMATION ENGINES.	62
TABLE 2-4. PROPERTIES COMPARISON OF UIDLS.....	64
TABLE 2-5. GENERAL FEATURES OF UIDLS.	65
TABLE 2-6. REVIEW OF EXISTING EVALUATION METHODS.	67
TABLE 3-1. CANONICAL LIST OF TASK TYPES.....	74
TABLE 3-2. TASK ITEMS.....	75
TABLE 3-3. POLYGONS APPROXIMATION TO A CIRCLE.	91
TABLE 4-1. MEDIATE USER INTERFACE ACTIONS.....	121
TABLE 4-2. QUESTIONS AND ANSWER CRITERIA TO SELECT A TOGGLE BUTTON.....	128
TABLE 4-3. WHICH WIDGET TO SELECT FOR WHICH ELEMENT OF THE DOMAIN MODEL?	129
TABLE 4-4. CORRESPONDENCE BETWEEN AIO TYPES AND CIO TYPES.	130
TABLE 5-1. TOOLS TO SUPPORT OUR APPROACH.	145
TABLE 7-1. CORRESPONDENCE BETWEEN AIO TYPES AND CIO TYPES.	178
TABLE 7-2. CONCRETE INTERACTION OBJET SELECTION.	195
TABLE 7-3. MAPPING FROM AUI TO CUI MODELS.....	209
TABLE 7-4. SHORTCOMINGS OF THE RENDERING ENGINE.....	223
TABLE 7-5. RESULTS FROM THE IBM CSUQ QUESTIONNAIRE.....	225
TABLE 7-6. STATISTICS COMPUTED TO ALL IBM CSUQ QUESTIONS.	225
TABLE 8-1. CONTRIBUTION TO UsiXML ONTOLOGY.....	235
TABLE 8-2. MODEL-BASED METHODOLOGIES CONTRIBUTION.	236
TABLE 8-3. COVERAGE OF THE EVALUATION METHOD PROPOSED.	237
TABLE C-1 THREE-D TOGGLE BUTTON EVALUATION TABLE.....	298
TABLE C-2 THREE-D OUTPUT TEXT EVALUATION TABLE	301
TABLE C-3 THREE-D INPUT TEXT EVALUATION TABLE.....	302
TABLE C-4 THREE-D COLOUR PICKER EVALUATION TABLE.....	304
TABLE C-5 THREE-D RADIO BUTTON EVALUATION TABLE	306
TABLE C-6 THREE-D CHECK BOX EVALUATION TABLE.....	308
TABLE C-7 THREE-D COMBO BOX EVALUATION TABLE.....	310
TABLE C-8 THREE-D BUTTON EVALUATION TABLE.....	314
TABLE C-9 THREE-D HYPERLINK EVALUATION TABLE	317

TABLE C-10 THREE-D MENU EVALUATION TABLE	319
TABLE C-11 THREE-D IMAGE COMPONENT EVALUATION TABLE	321
TABLE C-12 THREE-D VIDEO COMPONENT EVALUATION TABLE	323
TABLE D-1 COMMUNICATE USER INTERFACE ACTIONS EXAMPLES-----	324
TABLE D-2 CREATE USER INTERFACE ACTIONS EXAMPLES -----	325
TABLE D-3 DELETE USER INTERFACE ACTIONS EXAMPLES -----	326
TABLE D-4 DUPLICATE USER INTERFACE ACTIONS EXAMPLES -----	327
TABLE D-5 FILTER USER INTERFACE ACTIONS EXAMPLES-----	328
TABLE D-6 MEDIATE USER INTERFACE ACTIONS EXAMPLES -----	329
TABLE D-7 MODIFY USER INTERFACE ACTIONS EXAMPLES -----	330
TABLE D-8 MOVE USER INTERFACE ACTIONS EXAMPLES -----	331
TABLE D-9 NAVIGATION USER INTERFACE ACTIONS EXAMPLES -----	332
TABLE D-10 PERCEIVE USER INTERFACE ACTIONS EXAMPLES-----	333
TABLE D-11 REINITIALIZE USER INTERFACE ACTIONS EXAMPLES-----	334
TABLE D-12 SELECT USER INTERFACE ACTIONS EXAMPLES -----	335
TABLE D-13 START USER INTERFACE ACTIONS EXAMPLES-----	336
TABLE D-14 STOP USER INTERFACE ACTIONS EXAMPLES -----	337
TABLE D-15 TOGGLE USER INTERFACE ACTIONS EXAMPLES-----	338

Chapter 1 Introduction

1.1 What is a 3D User Interface?

Three-dimensional (3D) interactive systems offer significant opportunities for many areas of human activity – such as architecture, art and medicine –, and in computer science, such as videogames, information visualisation, Computer-Aided Design (CAD), and Human-Computer Interaction (HCI). Such areas usually involve training, simulation, spatial navigation, and exploration of complex data [Myer00]. Since many years, 3D interactive systems have demonstrated some benefits in reproducing adequately the reality [Shne03], in improving it, and even in augmenting it [Bimb05] by providing the user with unprecedented actions such as: undo, repeat action, group objects, or change properties.

In HCI, a *Graphical User Interface* (GUI) is often understood as a User Interface (UI) that involves graphical widgets that are displayed as planar regions in xy planes according to their abscissa and ordinates. As such, they are considered as *two-dimensional UIs* (2DUIs). Many desktop environments support overlapping of widgets, pseudo-relief effects (such as shadows) and depth effects, thus raising the level up to *two-dimensional and half user interfaces* (2D $\frac{1}{2}$ UIs) [Leac97]. In contrast, *three-dimensional user interfaces* (3DUIs) involve graphical widgets that are rendered as volumes in xyz spaces according to their 3D coordinates [Leac97].

Similarly to a 2DUI, a 3DUI can be decomposed into two parts: the presentation part (also called *front-end*) and the semantic core part (also called *back-end*) equipped with the semantic functions, the system data storage, and the communication layer. 3DUIs are often associated to various 3D systems, such as those in virtual reality, mixed reality, augmented reality, and 3D desktop environments [Lavi08].

2DUIs, as well as 3DUIs, could be used in order to support the user in order to achieve a 2D or a 3D interactive task. We hereby define a *2D task* as any interactive task that only require two dimensions to be conducted, typically xy . In contrast, a *3D task* may be extended up to three dimensions, typically xyz , where the last dimension is not necessarily spatial, but could be temporal, numerical, alphabetical, etc. A 2D task in nature, e.g., navigating on a map, can be supported by a 2DUI and does not necessarily require a 3DUI. A 3D task in nature, e.g., control-

Chapter 1. Introduction

ling a satellite in space, can be supported by both 2DUI and 3DUI, but not with the same quality. Table 1-1 categorizes UI types according to two axes: the nature of the task (2D *vs.* 3D) and the associated front-end (2D *vs.* 3D). For instance, navigating on a map, a 2D task in nature, can be rendered as a GUI on a 2D desktop, but also as a flat object in a 3D environment, although this is not particularly interesting. Organising tasks in windows, a 2D task in nature, is typically achieved in a 2DUI, but also benefit from a 3DUI (Figure 1-1). Controlling a satellite in space, a 3D task in nature, would really benefit from a 3DUI, although it could be projected into 2D planes as a GUI in 2D. In most situations, 3DUIs have shown some benefits with respect to 2DUIs, but also some shortcomings: they are respectively summarized in the next two sections. 3DUIs are not automatically superior or inferior to 2DUIs. Moreover, some transitions may become desirable from 2D to 3D and vice versa in order to ensure appropriate representation of a change of context and reasons exist why maintaining 2D contents in 3D [Delé09].

Front-end	Nature of the task	
	2D	3D
2D	2DUI, such as a GUI	GUI in 2D
3D	GUI in 3D	3DUI

Table 1-1. Nature of the task and its front-end.



Figure 1-1. TaskGallery, an example of a 2D task in 3D [Robe00a].

1.2 Why 3D User Interfaces?

Increase user satisfaction. 3DUIs are largely appreciated by specific categories of users and are helpful for specific tasks, but not all [Shne03]. In some circumstances, no significant difference exists in task performance between 2DUI and 3DUI. There is however a significant subjective preference for 3DUIs [Cock01]. 3D mobile maps surpassed 2D in performance and workload measures [Oula09].

Improve cognitive perception. Human perceptual mechanisms to analyze the world into structures of 3D primitives are better compared to 2D representations [Carr03]. Indeed, only a few objects can be held in our mind at the same time, and these are swapped in or out from either long-term memory or the external environment [Ande97, Kier97]. Object complexity is not as important [Kahn92] as object's structure. Object's shape is a primary element while colour and texture are secondary [Bied87]. A 3DUI of a UML class diagram improves the cognitive representation of the diagram with respect to a 2DUI [Iran00]. Stereo and motions cues are more effectively perceived in 3D than in 2D [Ware96]. Human visual bandwidth is much larger in 3DUIs than in 2DUIs [Shne02]. Users tend to remember better objects shapes and location in 3DUIs than in 2DUIs [Robe00].

Become a new deployment option. Since 1996, there is an increasing number of online communities in 3D virtual spaces: millions of users are already registered in such communities [BBCN06b], more organisations are increasing their presence on the web through these communities in order to sell their products and services [Bowm04], especially by social network applications. Consequently, 3D online contents are no longer the business restricted to videogames: non-gaming 3D applications would be prevalent in the short term [BBCN06a]. Accordingly to IBM, companies should actively venture into virtual worlds. IBM invested 11 million dollars in Second Life [Ryma08] to host virtual meetings, such investment has been rewarded with an increment in collaboration [Laud09]. 3DUIs will transform customer experiences, improve business processes, drive collaboration, enrich commerce and transactions, and enable 3D modelling and simulations so businesses can better understand their markets [Parr07]. Thus, 3DUIs become a new option to be considered when developing Information Systems (ISs).

Induce a sense of (tele)presence. 3DUIs are one manifestation of virtual reality scenes that may contribute to inducing, establishing, and maintaining a sense of user's presence or immersion in the scene [Slat99, Schu01], although this is not always the ultimate goal of the scene. When the virtual reality scene is manipulated remotely from the user's location, the effect is called telepresence.

1.3 Why not 3D User Interfaces?

Decrease user performance. 3DUIs do not necessarily increase user performance with respect to a 2DUI [Cock01], they sometimes decrease this performance [Sutc96, Cock02], in particular because 3DUIs raise issues such as: occlusion, layout management, and depth perception.

Increase manipulation complexity. 3DUIs, because of their very much nature, require extensive use of 3D input and output interaction techniques, such as physical devices for 3D and 3D displays. Mastering these devices and displays may require extensive training and/or proprioception capabilities [DeBo06b] that could go beyond the average skills of the user.

Are not appropriate for any task. Many information spaces, even those including spatial relationships, do not necessarily mesh well with 3DUIs, mainly because they do not assume any physical representation [Niel98]. Mapping a non-physical or a non-spatial relationship in a 3DUI represents an inappropriate use of a 3DUI. Even when a physical relationship is mapped into a 3DUI, the resulting 3DUI is not necessarily faster. For instance, moving physically in a 3D digital library is probably very enjoying, but less efficient in terms of navigation.

Are hard to evaluate for their usability. Bowman et al. [Bowm04] have argued that extrapolating usability guidelines, HCI heuristics from 2DUIs to 3DUIs may also obscure the specific characteristics of 3DUIs or even misaddress them: *"3DUIs are still often a 'solution looking for a problem.' Because of this, the target user population or interaction technique to be evaluated may not be known or well understood... Presence is another example of a measure often required in Virtual Environment evaluations that has no analogue in traditional GUI evaluation."*

1.4 Concerns

3DUIs are becoming the primary subject of interest of a growing community of researchers and developers [Dach06, Dach07 Lato08, Shae08] adopting different approaches for specifying and creating 3DUIs. Providing development methods and software support for 3DUIs is a complex problem. Researchers are at a stage where they are developing new interaction techniques, gestures and metaphors for 3DUIs [Myer00]. Most research and development is focusing on technological issues, as reported in a survey of major publications on 3DUIs [Swan05]. The research is mainly focusing on how to overcome hardware and software issues [Düns07]. Little or no attention is devoted to the design knowledge that should drive the development life cycle of 3DUIs. In this thesis, we hereafter identify a set of important concerns for developing 3DUIs for information systems:

Concern #1: Limited methodological homogeneity. Several methodological tools exist that can be organized into different categories depending on their goals, thus leading to a wide variety of methods for developing 3DUIs. They are used to involve interaction techniques, to describe 3D contents, to use different techniques, models, and specification languages.

Concern #2: Limited inter-method conceptual similarities to facilitate the abstraction of their models. Although these different methods share some similarities of their steps, several conceptual dissimilarities differentiate them. Consequently, it is challenging to transfer one abstraction from one method to another.

Concern #3: Limited methodological conceptual openness. Some methods possess their own underlying ontology that prevents us from extending it or transferring to another ontology.

Concern #4: Limited use of open specifications languages. When a method promotes any particular User Interface Description Language (UIDL), it does not necessarily mean that this UIDL is open to change, which is not a good guarantee that any evolution of the language can be implemented.

Concern #5: Limited explicitness in the transformations approach used. Transformations are in most methods hidden to the designer (i.e., built-in), untraceable and, not modifiable.

Concern #6: Single entry and exit points. Methods typically define their development process with one single entry point (i.e., the development process starts from an imposed artefact) and one single exit point (i.e., the artefact resulting from the development cycle is fixed by the method), without any modification.

Concern #7: Limited development methodologies for 3DUI information systems. The 3DUI development life cycle remains an art more than a principled-based approach. The methods reviewed rarely make explicit the design knowledge that is required for achieving each step. 3DUIs are not considered specifically for information systems.

Concern #8: Limited user-centred development methodologies. Development based on toolkits adopts a content-centric approach, as opposed to a user-centred approach. Hence, user involvement in the requirements analysis and evaluation are not formally part of the methodology.

Concern #9: Limited use of a consistent UIDL for 3DUI development. The reviewed methods are usually restricted to only one programming or mark-up language and do not allow easy porting from one to another.

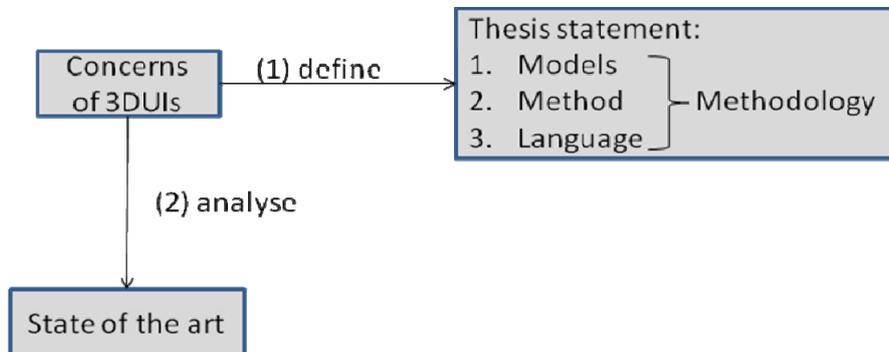


Figure 1-2. Benefits of the identified concerns of 3DUIs.

By addressing the aforementioned set of nine concerns, it is expected to benefit from the following results (Figure 1-2): (1) Contribute to define the thesis statement defined in Section 1.5.1; (2) Contribute to analyze a set of features of 3DUIs reviewed in Chapter 2.

1.5 Thesis

1.5.1 Thesis statement

In this thesis, we argue that developing 3DUIs is an activity that would benefit from the application of a **development method** which is typically composed of: (1) a set of **models** defined according to an ontology, (2) a **language** that expresses these models, and (3) a **principle-based approach** manipulating these models based on guidelines. Thus, we will defend the following thesis statement:

A model-driven engineering approach enables structuring the development life cycle of a Three-Dimensional User Interfaces of an Information System in a principle-based way.

The goal of this thesis is not to prove that 3DUIs are better or worse than 2DUIs regarding any particular metric, such as usability or presence. Rather, we take for granted results issued by the experimental studies mentioned in Sections 1.2 and 1.3: 3DUIs are not just more attractive for the users but also provide new opportunities for developers to manage information visualization [Cock01, Shne02]. In a review of applications, some examples of user performance in 3DUIs were identified [Cock01]: user preferences are on the use of 3DUIs, as users found them more natural to use, 3DUIs are easier to use for cognitive reasons, as it exploits the spatial memory and cognition of humans. Since not all tasks are appropriate for a 3DUI, the next section will motivate the focus of this thesis on some particular task types.

1.5.2 Focus of the thesis

1.5.2.a 3DUIs for Information Systems

3DUIs can refer to solutions addressing a large amount of various problems, such as 3D home pages, scientific visualisation, 3D chat rooms, 3D tours, and commercial visualization. The implementation complexity can range from a straightforward programming based on basic 3D graphics (e.g., in Virtual Reality Markup Language - VRML) to highly complex front- and back-ends, such as detailing the human body for real time neurosurgery. Facing this wide variety, this thesis will focus on Information Systems (ISs). Organisations conceive, develop and use IS to satisfy their needs. An *Information System* (IS) for an organisation is a construction made up of four blocks [Boda89]:

1. *Data*, a partial representation of facts that interest the organisation.
2. *Processes*, that represent means to acquire, search, store, present, and convey information.
3. *Organisation rules*, governing the implementation of informational treatments.
4. *Human & Technical Resources*, required for the functioning of IS.

An IS supports *management tasks* such as those depicted in a classic typology according two axes (Figure 1-3) [Boda89]:

- *Functioning level*, which ranges from operational, decisional to strategic.
- *Structure level*, which ranges from structured to informal.

In this thesis, we primarily consider management tasks that are operational and structured activities typically corresponding to administrative tasks, which are defined to deal with routine activities [Boda89]. In this scenario, a context of use is assumed to be *quasi-constant*: the physical environment is assumed to be an office setup, the user has known skills required to conduct these administrative tasks, and a desktop computer is considered as the main computing platform. Therefore, we consider that 3DUIs that correspond to other tasks than such administrative tasks are beyond the scope of this thesis, as well as contexts of use that significantly depart from this assumption. For instance, a mobile traveller could conduct administrative tasks, but the resulting context no longer satisfies our assumption. And so do 3D games, volumetric displays, Organic UIs [Holm08].

Assuming the administrative tasks and the context of use, it is crucial to select an *interaction style* that is appropriate. For this purpose, Table 1-2, respectively Table 1-3, consider interaction styles according to task attributes, respectively to user attributes. The focus on ISs induces the following values for these task properties:

Chapter 1. Introduction

- *Minimal to maximal prerequisites*, as the amount of knowledge required to the user to properly carry out the task with the intended UI varies. For instance, the prerequisites of an ATM should be minimal, whereas the UI for an air-traffic control system would surely be maximal.
- *Low to high productivity*, as the frequency of use varies depending on the task. For instance, a letter composition in an insurance company is of high productivity for insurance producers, whereas a monthly report is not.
- *Existent objective task environment*, whether an organisational task assumes the presence of domain objects.
- *Feasible environment reproductibility*, as the it is useful to represent domain objects as manipulable objects. For instance, IBM RealPlaces [Robe00b] reproduces physical objects that belong to the real world so that they can behave like in the real world (e.g., a phone system, a card filer).
- *Low to high task structure*, as the degrees of freedom or constraints that the user has in carrying out the task. For instance, calculating the roots of a second-degree equation is highly structured since a deterministic algorithm governs the process, whereas an advice-giving task for loans may reorder subtasks according to currently available information.
- *Low to high task importance*, whether a task in the organisation may be crucial or not. For instance, setting up an alarm in a control room is considered important, whereas editing a simple statistical report is not.
- *Low to high task complexity*, as the complexity degree of a task varies. For instance, a radar-tracking task is highly complex, whereas an advertisement composition is not.

The focus on ISs induces the following values for the user properties:

- *Task experience* (elementary, regular, rich): this parameter combines syntactic and semantic task knowledge. *Syntactic knowledge* refers to the task allocation and its position in the complete chain, including terminology, whereas *semantic knowledge* refers to domain objects, actions and procedures embedded in the task. If a user integrated these from both an intellectual and practical point of view, then the task experience is said rich.
- *System experience* (elementary, regular, rich): this parameter expresses the experience level required by technological means in order to carry out the task, such as printer facilities, file management, and word processing.
- *Task motivation* (low, moderate, high): this parameter translates the psychological user attitude with respect to the task. If the user is eager to carry

For administrative tasks supported by ISs, several interaction styles are candidates: form filling, multi-windowing, direct manipulation, iconic interaction, graphic interaction, multimedia interaction, and 3DUIs. 3DUIs were chosen as a potential interaction style for ISs since this option is underexplored. The task complexity and user experience are considered medium. IS examples that are not addressed in this thesis are: non interactive contents (e.g., a surgery room in Figure 1-4a), information visualisation (e.g., 3D statistics in Figure 1-4b), and custom 3D contents (e.g., a stadium in Figure 1-4c).

1.5.2.b About the Front-End

The last phase of a software development method, which is typically the implementation of a system, could be divided in two sub-phases: the UI programming (front-end) and the system functionality programming (back-end). The amount of effort in a 2D GUI has been reported to be important: the time for UI developing is estimated between 44% [Boeh88] and 90% [Myer92] of the total time dedicated to produce an entire system. 3DUIs probably do not escape from this observation and it is likely that they are increased. The 3DUI front-end is composed of a mixture of inputs and output *interactors* (i.e. objects that support interaction and task achievement) and *decorators* (i.e., any 3D static object that is present for visual purposes only). The stadium shown in Figure 1-4c is a 3DUI decorator since it does not consider any interaction, while the map is a 3DUI interaction since it is used to locate a seat, as well as the menu and the form that are used to purchase a ticket. Coupling the front-end with the back-end of an IS is not addressed explicitly in this thesis since it is largely covered in the literature [Bowm04].

1.5.3 Some Definitions

Model-Driven Architecture (MDA). The following definition was approved unanimously by 17 participants of the ORMSC plenary session meeting in Montreal on 23-26 August 2004. The stated purpose of these two paragraphs was to provide principles to be followed in the revision of the MDA guide:

”MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformation involved in MDA. MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modelling language must be used. Any modelling language used in MDA must be described in terms of the MOF language to enable the metadata to be understood in a standard manner, which is a precondition for any activity to perform automated transformation.”

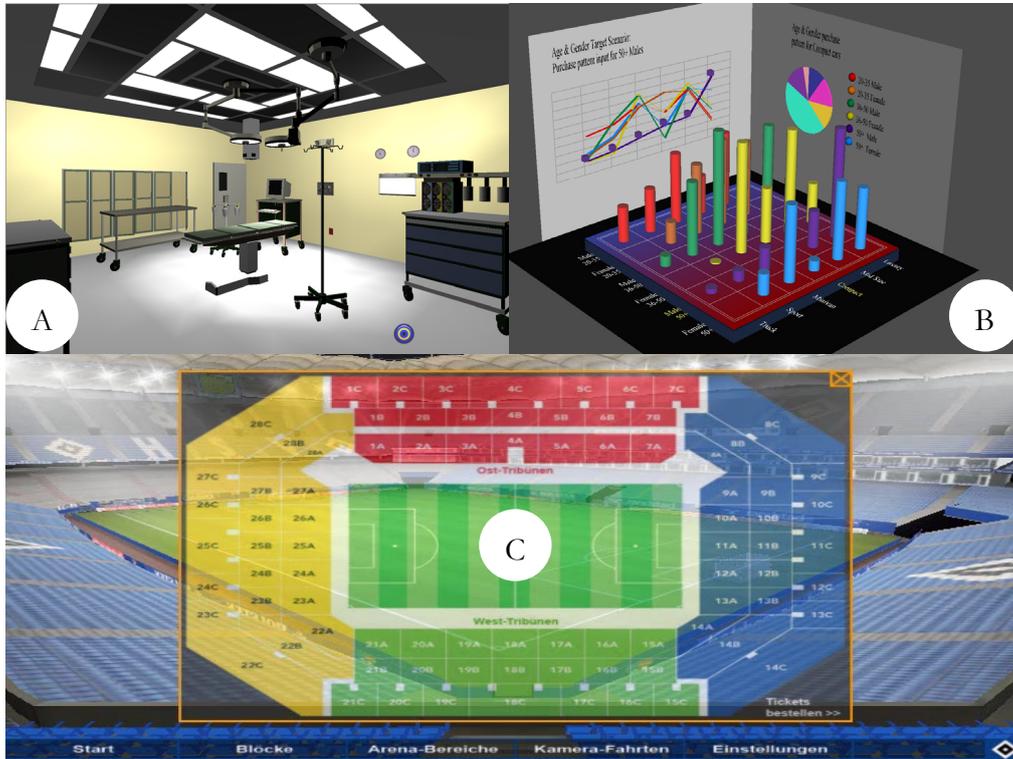


Figure 1-4. Some information systems out of the scope of this thesis: (a) a surgery room; (b) a 3D statistical visualisation; (c) a stadium seat booking system.

Notice that this definition emphasizes the transformations, the models, and the language. They all should come along with MDA compliant methods.

Human Computer Interaction (HCI). ACM defines Human Computer Interaction (HCI) as “*a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them*”. This definition attempts to cover the wide range of topics HCI covers, including disciplines: psychology, sociology. One of the concerns in this discipline is the development of User Interfaces (UIs).

1.5.4 Target Audience

A typical development life cycle is supported by steps in a development method. Such steps usually involve: system analysis, system design, programming, testing, evaluation, and maintenance. Several types of stakeholders may be involved in these steps, depending on project rules, constraints, and budget. Without going into details of the specific activities carried out in each development step or defining the roles of the different stakeholders. We have identified that the main beneficiary of this work is the designer. The designer is the stakeholder who is in

charge of creating design specifications based on the requirements. This role should benefit from the methodology devised in this thesis since it defines means to systematically design 3DUIs for ISs starting by task modelling. The task model is enriched and consolidated with links to a domain model that is also relevant to the designer. The designer is relying on her skills and experience to write these models that determine a 3DUI structured in an abstract way, i.e., independently of any interaction modality or computing platform. The designer can concretise such an abstract UI into a concrete 3DUI. Moreover, the methodology is enriched by a set of principles contributing on the design of the problem.

Of course the work of the designer and the principles that we proposed might be of benefit for the developer, and the end user. The roles of these three stakeholders are sometimes overlapping, if not merged, during the development life cycle due to the prior experience. For instance, an experienced developer could play a significant role in the system analysis and a designer could also program.

The *developer* is the stakeholder who is in charge of developing a final 3DUI from a concrete 3UI. The developer is relying on design knowledge for 3DUIs, such as: choosing adequate input/output devices and interaction techniques, representation of objects, selection of objects, manipulation of objects, arranging of objects, and usability guidelines for 3DUIs, such IBM RealThings design guidelines for 3DUIs [Mull98]. Relying on a predefined widget set of widgets in order to develop a 3DUI would certainly make the development process easier and faster than starting from scratch. This systematic development life-cycle may also reduce design creativity. The developer could benefit from previous development steps. Therefore, she may be involved in these steps as well. An abstract UI contributes to precise the logical, temporal, hierarchical relationships of a 3DUI.

The *end user* is the ultimate stakeholder who is expected to benefit from the method devised in this thesis. The end user could be characterised by several attributes such as, but not limited to [Guer08b]: capability, availability, task load, task experience, system experience, device experience, and task motivation. This information is useful for task allocation [Gonz09e]. Although this model contains attributes that characterises the end user, no particular action would be decided based on this information in an automated way since it is very sensitive, i.e. no 3DUI adaptation based on end-user experience or task frequency would be investigated since such an adaptation for 2DUI is already very challenging [Vand08], adaptation of 3DUIs is even more challenging. However, the developer must consider this information in the final representation of the 3DUI, the selection of the representation process is used for this purpose. The end user view must be reflected in the task model in some way. Consequently, the first step of our methodology

will consider end user involvement by means of a task model that will present the end user's viewpoint of the interactive system, whether it will be supported by a 3DUI or not. The 3DUI is targeted to the end user who is going to use it but also who could ultimately evaluate it. The methodology provides some means to support evaluation of 3DUIs based on heuristic evaluation and guidelines checking, thus requiring a definition of these usability guidelines.

1.6 Reading Map

This dissertation is structured into eight chapters that form the main body of the contribution, including this introduction (Chapter 1) and a conclusion (Chapter 8).

Chapter 2 reports on some significant pieces of work related to the paradigm of model-based development and 3DUIs. We discuss different approaches pertaining to the development of 3DUIs. A set of shortcomings is identified from a comparative analysis of such development methods. A list of requirements that addresses selected shortcomings is then decided that will help us to assess the appropriateness of the methodology devised in the thesis in Chapter 6.

Chapter 3 provides a conceptual modelling of 3DUIs that will be the base of the methodology based on viewpoints, capturing various levels of abstraction to characterise a 3DUI. An abstract syntax is then derived from this conceptual modelling and two concrete syntaxes will be drawn upon (i.e., graphical and textual). Finally, the language and the supporting software are described.

Chapter 4 exemplifies the fundamental principles that drive our model-driven transformational development of 3DUIs on three case studies sorted by increasing level of complexity. The first case study concerns the development of a 3DUI for an on-line polling system and is intended to facilitate the understanding of the method on a simple (but simplistic) example. The second case study concerns the development of a 3DUI for a salary management application for trainers and is intended to demonstrate the capabilities of exploiting the design knowledge at various steps of the method. The third case study concerns the development of a 3DUI for a Flight Management System (FMS) in an automated cockpit as an alternative to the existing character-based UI.

Chapter 5 discusses the selection of a UIDL, the reasons for such a selection, and how the conceptual modelling of 3DUIs has been achieved within this UIDL.

Chapter 6 presents the set of software modules that were used as support for the method through the different development steps. The software tools are briefly discussed.

Chapter 1. Introduction

Chapter 7 addresses the feasibility of the methodology by conducting three case studies selected in order to cope simple solutions and complex scenarios where 3DUI can be offered as an interaction style. An internal evaluation of the requirements is discussed with the results and the limitations of our research.

Chapter 8 concludes this thesis by assessing the appropriateness of the methodology with respect to the requirements elicited in Chapter 2, given the focus and scope decided in this introduction. This will also include a comparison with a similar method [DeBo06] in order to identify some open issues in the area. The theoretical, methodological, and software contributions of this thesis will be summarized and avenues for future work will be envisioned and analysed.

Appendix sections provides the reader with detailed information on which some parts of the main body of the thesis is based. Due to the size of some of the appendixes (A-B), a preliminary analysis is provided. Appendixes C-D give the complete details of some contents that are only summarized or briefly described in the main body of the thesis. Appendix E introduces a taxonomy of 3DUIs in which different components are used in a virtual application depending on the context of use, the nature of the task, and other parameters. In appendix F more details on the state of the art is presented. Finally, the appendix G task patterns are described.

Chapter 1. Introduction

Interaction style	Prerequi- sites	Productivity	Objective environment	Reproductibility	Task structura- tion	Task impor- tance	Task complexity
Command language	moderate	high	non existent	unfeasible	Low	high	low to moderate
Programming language	maximal	low	non existent	unfeasible	Low	low	moderate to high
Natural language	minimal	low	non existent	unfeasible	low	low	low to moderate
Function keys	minimal	high	non existent	unfeasible	low to moderate	moderate	low to moderate
Menu selection	minimal	moderate	non existent	unfeasible	moderate to high	low	moderate
Query language	moderate	moderate	non existent	unfeasible	Low	low	low
Quetions/Answers	minimal	low	Existent	feasible	High	low	low
Form filling	moderate	moderate	Existent	feasible	High	high	moderate
Multi-windowing	moderate	moderate	Existent	feasible	low to moderate	high	high
Direct manipulation	minimal to maximal	moderate	Existent	feasible	Low	high	high
Iconic interaction	Moderate	high	Existent	feasible	moderate	moderate	low
Graphic interaction	moderate to maximal	moderate	Existent	feasible	Low	low to moderate	low to moderate
Multimedia interaction	Minimal	low	Existent	feasible	moderate	low to moderate	moderate to high
3D User Interfaces	Minimal	low	Existent	feasible	low to moderate	low	low to moderate

Table 1-2. Expression of interaction styles in terms of task parameters [Vand00].

Chapter 1. Introduction

Interaction style	Task experience	System experience	Task motivation	Experience with modern interaction devices
Command Language	moderate to rich	rich	High	moderate
Programming language	Rich	rich	Rich	moderate
Natural Language	Rich	moderate	Low	high
Query language	Rich	moderate	Moderate	high
Questions/Answers	Elementary	elementary to moderate	Low	moderate to rich
Function keys	moderate to rich	elementary	Low	low
Menu selection	elementary	elementary	Low	low
Form filling	elementary to rich	elementary to rich	low to moderate	elementary to moderate
Multi-windowing	elementary	elementary	Low	low
Direct manipulation	elementary	moderate	Low	low
Iconic interaction	elementary to moderate	moderate	low to moderate	low
Graphic interaction	elementary	moderate	low to moderate	moderate
Multimedia interaction	elementary	moderate	Low	moderate
3D User Interfaces	elementary	moderate	Low	high

Table 1-3. Expression of interaction styles in terms of user parameters [Vand00].

Chapter 2 State of the Art

To identify shortcomings on existing work, a series of comparative analyses were conducted using the three axes recommended by Beaudoin-Lafon [Beau00]:

- *Descriptive part.* A common ground is needed to describe every piece of work.
- *Comparative part.* A set of criteria were defined to compare the different works that we described using a common syntax.
- *Generative part.* New work emerges from the comparative analysis as a result of the identification of limitations or potentiality of the literature review.

This chapter reports the descriptive part and comparative part for software support, methods, languages and transformation engines to support the development of 3DUIs. The conclusion summarizes these findings and ends up with a list of shortcomings and a set of requirements to be addressed in this dissertation.

2.1 Software Support

In the programmatic approach, the 3DUI is obtained by directly coding in its target computer language, e.g., C++. Nowadays, the programmatic approach is the more frequently used, particularly for applications where performance is a priority such as in games, the leading business for 3DUIs. Most games are written in C++ although, some may use C to try to get even more speed (at the cost of not having built in Object Oriented support) [Bake06a]. Therefore languages such as Java3D and C# are not used for mainstream games because they tend to run slower. This may possibly change in the future and factors like development time and the ability manage complexity may become more important. However, at the moment, there is not really a viable alternative to C++ for writing high speed games.

There are also some other application programming interface (API) that can be used in most common programming languages, for instance, OpenGL [Open04], the premier environment for developing portable, interactive 2D and 3D graphics applications, is the industry's most widely used and supported 2D and 3D graphics API.

Chapter 2. State of the Art

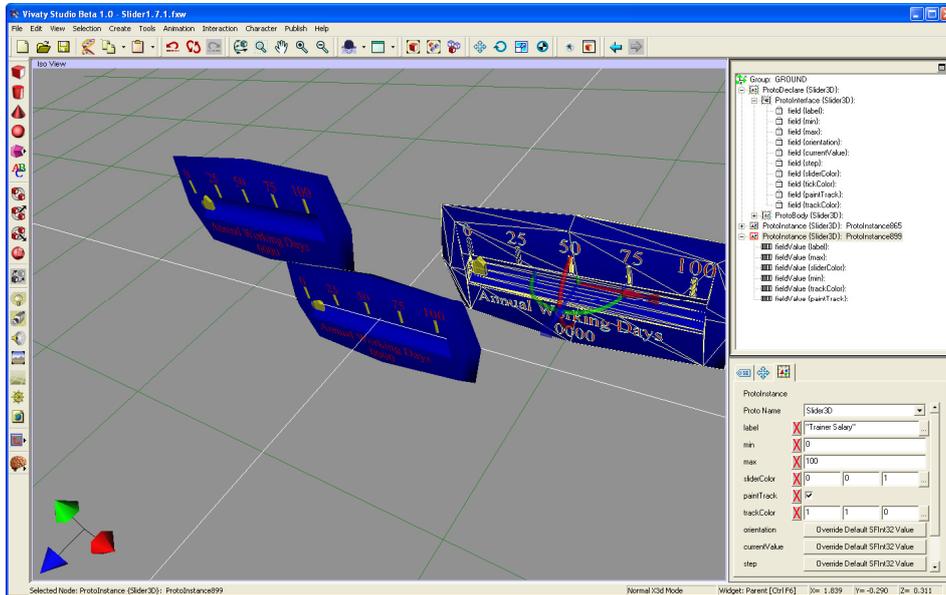


Figure 2-1. Slider design using Vivaty Studio (www.vivaty.com).

Recently Google launched its new API to create 3DUIs using JavaScript. The so called O3D (<http://code.google.com/intl/fr-FR/apis/o3d/>) is innovative in the sense that it was conceived for the web, which is a major step. However, it is still under development process and reaching a web standard status, its ultimate goal, will take some time.

Contrarily to the open source project from Google, Microsoft XNA is an API used for game development on Microsoft platforms. XNA is not just a framework like DirectX, it also contains a lot of tools and even a custom IDE derived from Visual Studio [Nits07] making easier the programming. XNA uses DirectX DLL making some aspects simpler, if we work with the DirectX SDK.

Nowadays, XML-based languages are used in a variety of applications. The description of virtual applications is not the exception. As a XML-based language could represent anything, a wide quantity of solutions has been proposed to 3D development, such as: X3D the VRML evolution. Even that there are more XML-based languages such as: InTML [Figu02] to describe virtual reality interaction techniques or VRXML [Cupp05, DeBo08] to define 3DUIs.

Toolkit 3D modelling is one of the most used mechanisms used to develop 3D UIs. In this case a toolkit is seemed as the set of basic building elements for graphical User Interfaces that can be implemented whether in a library, such as Open Inventor, or an application framework such as Vivaty Studio (Figure 2-1).

Table 2-1 sums up a comparative analysis of the toolkits reviewed. The properties analyzed in the comparison are:

- *Models* manipulated by the toolkit, if any.
- *Inter-mode linking*. Three objects were chosen in order to depict the different relationships:
 - **(A, ..., B)** indicates A, ..., B are grouped models that are done at the same level.
 - **A↔B** indicates that A derives B and B reengineers from A.
 - **A→B** indicates that B derives from A.
 - **A≈ ↔B** indicates that model A concepts could be manually linked to model B concepts and that B can be manually reengineered from A.
 - **A≈→B** indicates that model A concepts could be manually linked to model B concepts.
- *Target Languages* designate the languages of the target UI.
- *Availability of toolkits* (Avl. Mod.) Refers to the possibility for an external tool to process the manipulated software. Possible values:
 - **×**: not available.
 - **√**: open source or available at certain level.

The Cameleon Reference Framework (CRF) (Figure 2-2) considers four development steps [Calv03]:

1. *Task & Concepts* (T&C): describe the various users' tasks to be carried out and the domain-oriented concepts as they are required by these tasks to be performed.
2. *Abstract UI* (AUI): defines abstract containers (AC) and individual components (AIC) [Limb04a, Limb04b, Limb04c] two forms of Abstract Interaction Objects (AIO) [Vand93], that group subtasks according to various criteria (e.g., task model structural patterns, cognitive load analysis, semantic relationships identification), a navigation scheme between the container and selects abstract individual component for each concept so that they are independent of any modality.
3. *Concrete UI* (CUI): concretizes an AUI for a given context of use into Concrete Interaction Objects (CIOs) [Vand93], so as to define widgets layout and interface navigation. It abstracts a FUI into a UI definition that is independent of any computing platform. For example, in Envir3D [Vand04], the CUI describes traditional 2D widgets with mappings to 3D by relying on different mechanisms when such a mapping is possible.
4. *Final UI* (FUI): is the operational 3DUI, i.e., any 3DUI running on a particular computing platform either by interpretation (e.g., through a Web

browser) or by execution (e.g., after compilation of code in an interactive development environment).

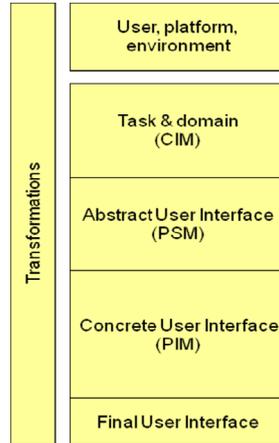


Figure 2-2. Cameleon-based methodology layered profile.

Note also in Figure 2.2 that T&C, AUI, and CUI correspond respectively to Computing Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM). Models that support each step are also important. Therefore, this review only describes the supported models at each step for specifying a UI. While the four steps have their corresponding models, there are some others to support a transition from one level to another:

1. *Context of use (C)*: is a model describing the three aspects of a context of use in which an end user is carrying out an interactive task with a specific computing platform in a given surrounding environment. Consequently, a context model consists of a user model, a platform model, and an environment model.
2. *Mapping model*: is a model containing a series of related mappings between models or elements of models.
3. *Transformation model*: is a model that defines the method used to support the model to model transformation.

2.2 Methods for 3DUI Development

So far, mechanism to produce, render, and visualize 3DUIs have been discussed. In this section, design techniques, methods, models and methodological frameworks are analyzed. There has been a long history and tradition to attempt to capture the essence of 3DUIs at various levels of abstraction for different purposes. The return of this question today gains more attraction, along with the dissemina-

tion of languages, tools that gives birth too many proposals for a new 3DUI development methodologies. Consequently, there is a need to conduct an in-depth analysis of features that make all these proposals discriminant and appropriate for any specific purpose. The review is conducted on a significant subset of such methodologies on an analysis grid and UIs within the scope of this thesis.

In order to have a common ground a Model-Driven Framework has been chosen. In software engineering, specification-based (or model-driven) approach relies in the power of models to construct and reason about software systems. This approach is based on models. In order to generate them, the main properties of real objects must be identified. For this purpose, some kind of judgment is required.

The goal of Model-Driven Approach, for UI development is to propose a set of abstractions [Stee08], development processes, and software tools enabling a engineering approach of UI development. The characteristics of an engineering approach are its systematic (development based of rational principles), its reproducibility, its orientation towards quality criteria.

2.2.1 Web & Information System Engineering Lab

The virtual reality Web & Information System Engineering (VR-Wise) [Pell05b] project looks for conceptual modelling of Virtual environments. Starting from web design methods and design methods for Virtual Reality. WISE Lab has three main project aimed at designing and specifying Virtual (Web) Environments in a systematic way and that can be supported by design methodologies and tools.

They have developed models for virtual objects in OntoWeb tool [Pell04a], part of the *OntoBasis* project. They called their models ontologies and with them they look for Foundations, Construction, Services and Applications. Their approach is based on the idea that as a Virtual Environment is composed of objects it may be possible to extract properties of these objects and their relationships in the Virtual Environment from available ontologies covering the domain under consideration.

From the ontologies they build virtual correspondences mapping the ontologies concepts one-to-one to virtual properties, which is a good first step to design Virtual worlds. Considering our methodological development path we place this approach at the Content description for UI, in which objects/content are described in models that correspond to a specific platform. They generate a direct mapping from the concepts described in the domain model to virtual objects VRML or X3D.

2.2.2 CoGenIVE

CoGenIVE (Code Generation for Interactive Virtual Environments) is a tool-supported process developed at the Expertise centre for Digital Media (EDM), a research lab at Hasselt University. The tool has been created in order to support and evaluate a model-based development process (depicted in Figure 2-4), to facilitate the creation of multimodal IVEs [Cupp05, Cupp06, DeBo08].

The first explicit artefact in the development process is a Task Model, expressed in ConcurTaskTrees (CTT) [Pate97]. This notation uses a graphical syntax and offers both a hierarchical structure and support to specify temporal relations between tasks. Four types of tasks are supported in the CTT notation: application tasks, user tasks, interaction tasks, abstract tasks. Sibling tasks on the same level in the hierarchy of decomposition can be connected by temporal operators.

Once the Task Model is created it will be converted to a Dialog Model, denoted as a State Transition Network (STN). The STN is based upon Enabled Task Sets (ETS), which can be automatically generated from the CTT by means of the algorithm described by Luyten *et al.* [Luyt03]. Each ETS consists in the tasks that can be executed in a specific application state. Since we are developing *Interactive Virtual Environments*, we will only elaborate on the interaction tasks within the application. A possible example of such a task is object selection. In a typical form-based desktop application, selecting an item is a straightforward task in which the user selects an entry from a list. In an IVE however, this task becomes more complex because of the many selection metaphors (e.g., touch selection, ray selection, go-go selection).

To handle this problem, an Interaction Description Model, called NiMMiT, has been created. The goal of NiMMiT is to describe an interaction task in more detail. The diagrams can be created by means of the CoGenIVE tool and are exported to an XML file which is loaded at runtime. The connection of the diagrams to the interaction tasks in the dialog model is currently done by hand. A more detailed description of NiMMiT, together with some examples, can be found in [Vana06].

Within CoGenIVE the user can create user interface elements such as dialogs, menus, and toolbars that are then expressed in a VRXML presentation model. VRXML is an XML-based UIDL, so that the resulting resources are loaded into the application at runtime as well (Figure 2-3). VRXML examples and a motivation for the creation of this UIDL can be found in [Cupp04]. Like the interaction descriptions, the user interface elements should be connected to the different application states manually.

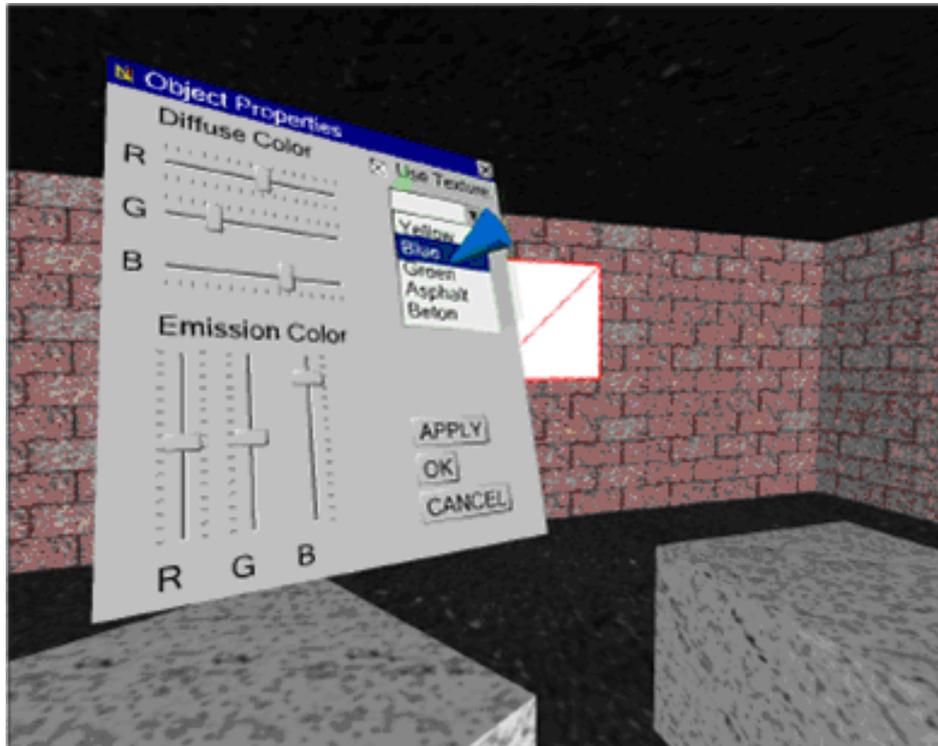


Figure 2-3. VRXML object property dialog [Cupp05].

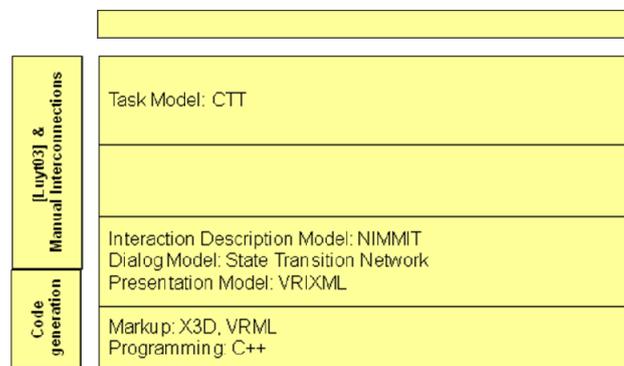


Figure 2-4. CoGenIVE development process.

Once all models have been created and connected they are used to automatically generate a prototype of the IVE together with the external resource files in which the NiMMiT and VRXML descriptions are stored. This approach offers some extra flexibility since part of the application behaviour can be changed without regenerating the code of the virtual environment. CoGenIVE covers several of a MBD methodology. The process starts from a task-model and incrementally evolves towards the final user interface. The first increment (towards the dialog model) can be done by an automatic transformation. Afterwards the designer has

to manually connect the presentation and the interaction description model. Preservation of manual changes is conserved only in the second transformation, resulting in possible inconsistencies between models that are manually adapted. However; once the code is generated from the designed models, manual adaptations are tracked and saved. This way, when regenerating the application prototype, the manually inserted code is preserved.

Preliminary evaluation of the CoGenIVE process in some IVE realizations has shown a considerable reduction of development time. Currently we are working on a new version of CoGenIVE with improved and more integrated tool-support.

2.2.3 Interaction Techniques Markup Language

The Interaction Techniques Markup Language (InTML) methodology [Figu04] is aimed to design VR applications. It enables the designer to concentrate on the architecture of the application, without dealing with too many details. The filter entity, the abstract building block of InTML, can be any device, interaction technique, behaviour, or content in a VR application. Its interface is defined in terms of input and output ports (IPort and OPort), 28 events are handled for them.

The behaviour description corresponds to the dialog specification, objects and devices models are part of a concrete model, because it is independent of implementation, at first view, assuming that there is no direct relation between those components and the implementation. The mechanism of InTML start with the specification of the goal of the projects, with the definition of the main tasks, everything is documented in InTML documents, which are refined through the process [Figu04]. This task is done manually without using standard guidelines for software elicitation. For this reason task is something that injects knowledge.

Similarly as tasks, the code generation for the FUI is straight forward of InTML methodology. At the CUI level all concepts are encoded using InTML [Figu02], a XML-based language for defining VR content, especially for Interactions Techniques. InTML is a black box for details about gathering information from devices or about object behaviour, as is “described” with code of programming languages. Also, geometry or other media types related to VR objects are produced in any of the available tools for that purpose, such as Maya, 3D Max, or Blender. While dataflow-based languages such as VRML focuses on description of geometry and animation, InTml focuses on the integration of application-specific behaviour, object behaviour and events from input devices, which is a tedious task in VRML, less complicated actually in X3D. Geometry is something that is described at a lower level, in a loadable format, and InTml refers to it as a reference to an object. The same can be applied to sound or haptic content.

2.2.4 Contigra

COmponent-orieNted Three-dimensional Interactive GRaphical Applications (CONTIGRA) project CONTIGRA [Dach02] is a XML application on top of X3D that give users components for interaction techniques and control widgets. The project has models for behaviour, Behavior3D [Dach03]. The core concepts in Behavior3D are animations, sequences of actions, and state machines. On top of them, other behaviours can be defined. Behavior3D targets 3D applications running over the web, in standard PCs, due the intrinsic characteristics of X3D and its interaction model, [Figu04]. Audio3D [Hoff03] is a solution of offered to describe 3D applications with audio that allows the description of complex acoustic environments but is still suitable for efficient real time sound rendering with 3D sound APIs. Similar to X3D a hierarchical, acyclic scene graph is used in Audio3D to organize nodes in groups and subgroups. Contigra concepts are mapped directly to a FUI, as they are described using X3D profiles. So Contigra development process is composed of dialog (Behaviour 3D), Concrete UI (Models involved for defining the UI) and Final User Interface is generated automatically.

2.2.5 TRES-D Methodology

The TRES-D methodology [Moli08] is the closest work to this dissertation. It considers a set of steps for the development of 3DUIs and provides a number of models. The 3DUI is defined by combinations of objects, including: avatars, agents, reactive objects, dynamic objects, static objects and processes. An object is an aggregation of geometry, appearance, perception and functionalities. The Interaction model involves the concepts: dialogue, tasks, operations, interaction techniques, actions, controls and physical devices. A distinction is made between high-level tasks and sub-tasks, and within the latter the lowest level ones are the operations, along with the required information units. The leaves are the Basic Interaction Tasks (BITas), the other being Composite Interaction Tasks (CITas).

A distinction is also made between interaction techniques and controls. A control, e.g. a 3D widget (3D representation of 2D GUIs), is no longer considered as an interaction technique here, but only a part of it [Moli08]. These controls are distinguished, similarly as task, in Basic Interaction Techniques or BITes, and Composite Interaction Techniques or CITes. The Space meta-model distinguishes between the 2D digital space, the 3D virtual space and the 3D real space.

The methodology, as described in [Moli08], starts with the problem definition. The second step, or detailed study, is aimed at developing such solution up to its final deployment and further maintenance. Between both phases mediates and

Chapter 2. State of the Art

agreement with the client, who has to approve the proposed solution. Before carrying out the design it is necessary to understand what is meant to be accomplished, task that is performed by analysts. At the end of this stage we want to know what kind of application is to be developed, what are the characteristics of the people who will use it, and what tasks will be carried out by them.

A task and domain analysis shapes a set of scenarios and highlights spatial relationships. A group of designers works on different solutions for the application. Artists, digital content creators, programmers, domain experts and users participate too. The main aim of this stage is to envision a solution that solves the problem, satisfies the user and, at the same time, is feasible. At the end of the stage, a complete report is written describing the solution and detailing its benefits, time and cost of development, risk assessments, so that a proper decision can be made. At the second phase, the design is done at two different levels: one detached from the implementation details and another one tightly related to that implementation.

A summary of the method using our comparison protocol is presented in Figure 2-5. Similarities with our work can be summarized to: we use the VUI toolkit as one possible output at the FUI level; the taxonomy used is a common agreement resulting on the collaboration with the author of this methodology. However, there are differences, including: the models, the approach, transformation system, abstractions the language. For instance, limiting 3D widgets definition limited to a 3D representation of a 2D GUI this methodology does is restrictive to what is expected in 3D which is to have any representation of real life objects. This is one of the differences we have for controls in a 3DUI.

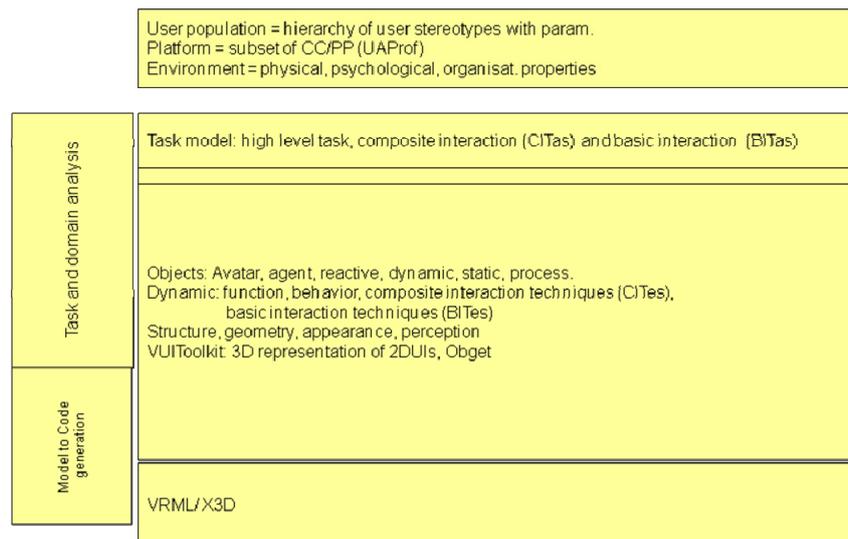


Figure 2-5. TRES-D Development Process.

2.2.6 Desktop Based Methods

3DUIs are now part of the desktop platform. This dissertation focuses on such platforms rather than those including sophisticated hardware and complex graphics. Still, the desktop platform must consider the interactive nature of virtual worlds and the need to keep a balance between performance and usability. Performance is a characteristic linked to the optimization process for a desktop 3DUI. This includes the Level of Detail (LOD) of objects, which can be adjusted based on the performance of the graphics hardware. That reference should be elicited in a plan or design stage previous to construction. The second issue, usability, is addressed by following guidelines at the design level [Kaur98]. In [Kim98], [Smit01] and [EON], the process for creating 3DUIs just considers the CUI model where implementation Modelling objects and Edit behaviour using either a script editor or a software tool to store models in files (VRML, X3D).

In [Hay] and [Daly02]: a very detailed method for 3DUIs is described. This approach explains the process considering the models and abstraction of the VRML and X3D as language target. The process starts with an informal project documentation that opens the door for planning using a storyboard to identify and differentiate interactive from content objects. The process of implementing the identified objects includes defining: shape that can be simple or complex, animation, shading, lighting, navigation methods, appearance, textures, sound, image, video, colour and set viewpoints. Last step is to assemble all the objects to build the 3DUI and publish it on the web using the language X3D or VRML.

A more robust method proposed by [Kaur98], which is more detailed and used in our methodology, Chapter 3, and for the evaluation, Chapter 7. The methodology starts with requirements documentation. This documentation derives a 3DUI mock-up. In order, to be as realistic as possible, it is recommended a collection of Data from Real World corresponding to the phenomenon of study. The way to implement the 3DUI depends on idea generation, resulting from user and group brainstorming meetings. This assumes that groups of experts might be available to design the 3DUI. Objects are modelled considering: lighting, appearance, textures, sound. Then the 3DUI is the result of assembling VE by putting together all the objects. The final step is to evaluate the resulting 3DUI.

Task modelling is not part of these methods, neither the AUI model. The second seems to be normal as the target language is known and is not independent of the modality of interaction, it is clear that is graphic. Most of the concepts that are, by the way, are fully reused for the CUI model is a very detailed abstraction of the characteristics of a 3DUI.

2.2.7 Participative Method

Methods of the previous groups assume that just one developer carries out the development. It also assumes that they know well the content of the virtual world or the preferences of the future user. Participative methods promote a higher involvement of both the user and the domain expert in the development, in order to obtain a product that satisfies the user, and with the right content.

In [Cele01a][Cele01b] a 3D content-centred methodology considers: Requirements of interaction; problem specification, including: content identification; Implementation: Content assemble; Evaluation: User experience. Similarly [Neal01] in their user-centred development process proposes a problem specification document. That is used for the planning of the 3DUI, including: idea generation, where future users are involved in working groups gathering ideas resulting from brain storming sessions and discussion. A storyboard is created with experienced developers that review the first draft of ideas. The next step is the implementation of the system. The final result is evaluated both with end-users and experts. In both methods the user experience is evaluated. The result is used as feedback for the next iteration of the method where the design is reviewed.

2.2.8 Task analysis for building VEs

As opposed to previous methodologies, a user-centred approach relies on a task models. In this category, mainly task analysis on task models is performed to evaluate the feasibility of the proposed solution in terms of performance. Some authors just described their methods in terms of evaluation mechanisms, for instance, task analysis used at the requirements specification level in [Gabb99; Pere00; Garc01; Poly05]. The problem specification considers conception design, human computer functionalities, and interaction techniques selection [Pere00; Garc01; Poly05]. Finally, just [Gabb99] considers evaluation on user experience based on experts' guidelines and heuristic evaluation.

Two 3DUI methodologies introduced by [East01; Wils02] at the requirements specification level an environmental model (Target platform) is specified where technological limitations, constraints and objectives of the project are established. The problem specification, includes: definition of interactions, any action on the part of the user that results in a change in the 3DUI. Also, points of interaction definition, which objects: Cues for interaction: none or small, hidden, obscure, colour coded, method for selection, desired effect of interaction, main feedback: visual, audio, none, availability of interaction if object is visible: always, sometimes, never, and feedback if interaction is available: none, textual prompt.

In addition, the navigation type considering the constraints of the system, including: Interface (Input physical devices, control characteristics and Output, physical devices, HMD, headphones) and the environment (fidelity of the representation and possible enhancements). The implementation consists on assembling the objects, what is called the topography of the 3DUI. The final step is the evaluation of the 3DUI, where the fidelity and validity of the 3DUI is evaluated. Also usability evaluation on the user experience is performed, including: usability, attitudes and side/after effects (physiological and psychological effects).

The main problem found in these approaches is the lack of model to model transformation guidance. No explicit rules or mechanism are detailed. It is then assumed that the developer or designer is experienced and is capable of using these methodologies.

2.2.9 Comparative Analysis on Model-based Developments

In this review of the literature (Table 2-2), the evaluation method used in each different methodology is compared. The Cameleon Reference Framework (CRF) [Calv03] steps are again considered. The properties analyzed are:

- *Models* manipulated by the methodology.
- *Inter-mode linking*. Three object were chosen to depict the different relationships:
 - **(A, ..., B)** indicates that A, ..., B are grouped models that are done at the same level.
 - **A↔B** indicates that A derivates B and B is reengineered from A.
 - **A→B** indicates that A derivates B.
 - **A≈↔B** indicates that model A concepts could be manually linked to model B concepts and that B can be manually reengineered to A.
 - **A≈→B** indicates that model A concepts could be manually linked to model concepts B. This means that the rules and the models exist but not a tool to support the automatic transformation.
- *Target Languages* designate the languages of the UI to produce.
- *Availability of models* (Avl. Mod.) refers to the possibility for an external tool to process the manipulated models. Possible values:
 - **✗**: models are stored in an internal format not made explicit e.g., models are tightly coupled with the tools.
 - **✓**: means that an external format for models exists e.g. models are available under a machine understandable format. A typical form is an XML-based language.

- *Extensibility of models definition* (Ext.Mod.) refers to the ability of extending definitions of models with new elements.
 - **Orig.:** means that models were intended to be extensible, but only by the originator of the methodology. This guarantees some interoperability of tools around a language.
 - **Design:** means that models are extensible and the designer (e.g., the tool user) is responsible for this extension.
 - **✕** : means that no mechanism supports model extension e.g., the system is bundled with a particular set of model definitions.

2.3 Transformation Systems Comparison

We conducted a comparative analysis [Gonz08a] on transformation engines. Our analysis was based on three existing tools TransformiXML [Limb04e], AGG [Erme99] and AToM³ [deLa02]. Even more a third custom transformation engine was build and compared, YATE (Yet another Transformation Engine) [Dela07]. The selected criteria (Table 2-3) were chosen considering the key factors that are relevant for: (1) implementing of a transformation engine (implementation paradigm and required programming skills, code generation support, pattern matching API), (2) usability (rules organization, rules scheduling organization) and further use of the transformation engines (maintainability, flexibility, and completeness).

Implementation paradigm and required programming skills. First major difference between the transformation engines is whether they are *imperative or declarative*. Like all graph transformation tools, AToM³ and AGG are strictly declarative while YATE and TransformiXML are imperative. *Performance* is a very important criteria and AToM³ posses a major disadvantage on this aspect. As AToM³ is entirely coded in Python and compiled at execution the execution is slower compared to precompiled Java code, YATE, AGG and TransformiXML. Moreover, the fact that the transformations are graphically designed means that the user does not have the possibility to optimize how they are executed, the engine is in charge of that. YATE and TransformiXML are fasters during execution. Not only because the compiled code is faster that the interpreted code of AToM³ but also the programmer chooses exactly how the transformation will be executed. *Redundancy* can be eliminated as well by grouping transformation rules. While AToM³ needs programming skills, as there are a few things to code in Python, like conditions, actions, constraints and variables' valuing, AGG requires programming skills to specify conditions on attributes. However most of the transformation are described graphically in AGG and AToM³, consequently they are more flexible because even GUI designers with (almost) no programming skills can use them, and

also because maintaining the rules is simpler. Nevertheless, in AGG and AToM³, the transformation rules are specified for the meta-model that we have graphically created and a change to this meta-model will make the transformations not work anymore.

The model-to-model approach. TransformiXML, AGG and AToM³ use graph transformations. The models are created graphically and so are the transformation rules. This is a very intuitive and easy to read and maintain. YATE uses another approach. Instead of transforming the model into a graph, it directly modifies it. In fact, the XML file is read and transformed into Java objects. These objects are then modified by the transformation rules and finally the Java objects are translated back into XML. In YATE, a method is created for each transformation rule. Comparing with the graphs the code is harder to read and maintain.

Code Generation. In YATE, code generation is feasible; the Castor API supports this process. On the contrary, AGG and AToM³ code generation is possible but it is poorly developed. To generate XML code for a model in AToM³, we should use transformation rules, with the “action” code writing XML code to a text file. This would be long and tedious, if possible.

Pattern matching. This is a big difference among all the tools. While pattern matching is supported by TransformiXML, AGG and AToM³, YATE does not support it for the moment. Pattern matching is still usable in YATE, but with the help of external projects or at the price of a long a fastidious implementation. This is more complex to implement than in TransformiXML, AGG and AToM³.

Rules scheduling and organization. AGG and AToM³ only allow determining a fixed order on the rules. There is no possibility of explicit flow control on them. And there is not a “call” instruction for a rule to call another. In fact, preconditions serve to decide whether or not the rule will be executed. In AToM³ it is possible to create distinct set of rules and execute one after another. However, there is no rule inheritance mechanism and import mechanism either. Therefore rule sets cannot import a rule from another set and a rule can neither use another. Finally, TransformiXML and YATE, allow rules organization (for example one class for each rules set), and the flow control is of course explicit. In YATE it is possible even to call a rule in another, because each rule is a method in Java.

Flexibility and maintainability. The more readable the models the easiest to maintain them. AGG, AToM³ and TransformiXML offer a best capacity to modify and maintain meta-models, models and transformation rules. However, when modifying rules at least two issues must be taken into account: 1) if a meta-model is modified then transformation rules designed for that meta-model are potentially

not working anymore (because they can apply on objects removed or modified); 2) modifying rules is sometimes difficult, for instance in ATOM³ modifications on rules leads to unexpected behaviour. Finally, because of its fully programmatic approach and more complex syntax, YATE is the most difficult to maintain and modify, as it implies several modification in several classes in addition to coding the transformation rules. So, flexibility is worse than for the two other three tools. *Completeness* refers to the ability of the system to handle complex rules and generate code. As ATOM³ allows programming in Python, it is able of executing more complex rules than strict graph-transformation rules (with only NAC, LHS and RHS). Still, the lack of explicit flow control and the absence of imperative constructs limit it. Because the source and target model are not distinct, they both obviously can be navigated and modified. Finally, the lack of code generation makes ATOM³ less complete. Finally, in YATE the limit is in fact the programming skills of the graphical interface designer.

Finally, Figure 2-6 a subjective interpretation of variations between completeness and performance of transformation engines which is desired to be as high as possible. HCI is a discipline particular with constant changes so it can be said that incompleteness is intrinsic to HCI. However completeness on existing tools is still relevant and for UI development transformation engines seems that TransformiXML and YATE are more complete compared to AGG and ATOM³. This conclusion is reinforced by the fact that AGG and ATOM³ are general purposes transformation engines which are hard to adapt to the UI development process. On the other hand, the limited support for maintainability due to the inflexibility contrast with the other tools. The general purpose engines are more easily to maintain compared to the specific purpose tools. Finding equilibrium is being around TransformiXML but selecting a transformation engine depends on the particular needs of the development team.

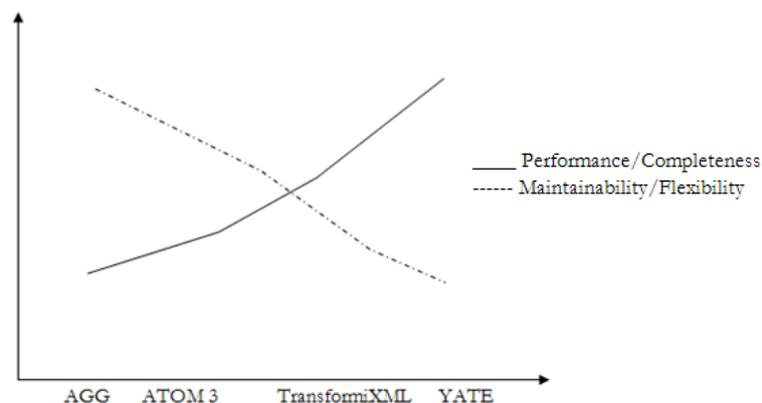


Figure 2-6. Tools comparison with respect to performance and maintainability.

2.4 User Interface Description Languages

For years, HCI witnessed a perennial race for the ultimate User Interface Description Language (UIDL) that would ideally capture the essence of what a UI could be or should be. A UIDL consists of a high-level computer language for describing characteristics of interest of a UI with respect to the rest of an interactive application. Such a language involves defining a syntax (i.e. how these characteristics can be expressed in terms of the language) and semantics (i.e., what do these characteristics mean in the real world). It can be considered as a common way to specify a UI independently of any target language (e.g., programming or markup) that would serve to implement this UI.

The issue of UIDL was first raised when it was required to develop a UI like a module of an interactive application rather than merely a series of lines codes. In a second time, the issue was reinforced when the desire appears to model a UI by a set of specifications so as to communicate these specifications and to share them across stakeholders. Or to (semi-)automatically generate the code of the UI, as desired in model-based approaches for developing UIs. When a UI is required on different computing platforms, this need took shape in some language that would be exchanged from one platform to another without any changes.

For some years, the race progressively slept. The wide availability of markup languages and the capability of introducing any language based on XML meta-language, along with the multiplicity of today's available platforms (e.g., mobile phone, smart phone, pocket PC, handheld PC, Tiquit PC, tablet PC, laptop, traditional PC, and even wall screens) have awakened this race and have exacerbated it to a point where today more than a dozen of UIDLs exist that focus on some of the desired characteristics. To shed light on this proliferation of UIDLs, we conducted a systematic comparison based on an analysis grid. The paper focuses only on XML-based languages, because XML is a well established standard that is easily extensible and that could work with yet-to-be-invented appliances without many changes. Furthermore, it is declarative and can be use by non-programmers.

In order to identify the main aspects of some of the most relevant UIDLs a comparative analysis was conducted [Guer09b]. For the purpose of the survey, we gathered and analyzed as much literature as possible on each UIDL. Then, depending on available tools, we systematically developed a multi-platform or multi-context UI for a simple dictionary so as to identify the capabilities of the UIDL and the ability of this UIDL to be supported by editing, critiquing, analysis tools, and, of course, tools for producing runnable UIs, both by compilation/execution and by interpretation.

Many UIDLs have been introduced in the literature and are widely used in practice. With this comes need to understand their scopes and their differences. The protocol selected was first used on a previous review on UIDLs [Souc03]. Table 2-4 compares the properties of the different UIDLs according the eight criteria:

- *Component models*: this criterion gives the aspects of the UI that can be specified in the description of the UIs. The *task model* is a description of the task to be accomplished by the user; the *domain model* is a description of the objects the user manipulates, accesses, or visualizes through the UIs; the *presentation model* contains the static representation of the UI and the *dialog model* holds the conversational aspect of the UI.
- *Methodology*: different approaches to specify and model UIs exist: 1) Specification of a UI description for each of the different contexts of use. As a starting point, a UI specification for the context of use considered as representative of most case, the one valid for the context of use considered as the least constrained or finally the one valid for the context of use considered as the most comprehensive is specified. From this starting UI specification, corrective or factoring out decorations [Souc03], e.g., to add, remove, or modify any UI description) are applied so that UI specifications can be derived for the different contexts of use. 2) Specification of a generic (or abstract) UI description valid for all the different contexts of use. This generic UI description is then refined to meet the requirements of the different contexts of use.
- *Tools*: some of the languages are supported by a tool that helps designer and renders the specification to a specific language and/or platform.
- *Supported languages*: specify the programming languages to which the XML-based language can be translated.
- *Supported platforms*: specify the computing platform on which the language can be rendered by execution, interpretation or both.
- *Abstraction level*: each UIDL may exhibit the capability to express a UI (instance level), one or many models involved in the development of this UI (model level), how these models are built (meta-model level), and what are the fundamental concepts on which this is based (meta-meta-model level).
- *Amount of tags*: to reach the above level of abstraction, each UIDL manipulates a certain amount of tags.
- *Coverage of concepts*: depending on the level of abstraction, each UIDL may introduce some specific vs. generic concepts (e.g., a given presentation model vs. any model, each custom-defined), their properties (e.g., to what extent can a concrete presentation be specified), and their relations.

The same protocol used for the methodologies comparison was used to compare the features of existing UISLs, including the concepts and transformation applied. Table 2-5 compares UIDLs along the following dimensions:

- *Relatedness* specifies if the language is associated to the development of UIs.
- *Standard* specifies if the UIDL is already a standard language.
- *Specificity* indicates if the UIDL could be used in one or many platforms.
- *Publicly available* specifies the analysis depth. Depending on the availability of the language deep analysis can be done. This category was used to discard many languages that lack on documentation or that is confidential. The possible values are: 0 = no information available, 1 = not available, 2 = poorly available, 3 = moderately available, 4 = completely available and 5 = completely available with meta-models.
- *Level of usage*: depending on the usage of the language we create the following categories: 0 = unknown, 1 = one person, 2 = two or more persons, 3 = one organization, 4 = two or more organizations and 5 = massive usage.
- *Weight of the organization* behind the UIDL denotes who is belonging, promoting the UIDL.
- *Type* specifies whether the UIDL is a research or industry work.

2.5 3DUIs Evaluation Methods

Some authors of 3DUI methodologies introduced usability evaluation in intermediate development steps. Some others just evaluate the final result. There is a wide set of techniques used for evaluating 3DUI. Those techniques have been proved to be good not just in immersive applications but also in other context of use, such as: desktop UIs, Web UIs. If fact none of exiting tools and methods used in 3DUI evaluation are new or unique [Bowm04], they have been used for 2DUI evaluation. A set of methodologies for 3DUI development were selected, analyzed and gathered. There is a plethora of methods and we may have skipped some significant work. However, our goal is not to be exhaustive.

Wilson, Eastgate & D'Cruz [Wils02] introduced different actions at early development steps to evaluate usability: Feedback definition (this characteristic is used to reinforce the comprehension on the 3DUI); then, a survey of Interaction Techniques (IT) is needed. Based on the selected IT, task analysis is performed over task models to evaluate speed and applicability of the 3DUI. Once implemented, the 3DUI is evaluated to check: the fidelity, compared to the real life stuff it represents; and the validity, if the 3DUI corresponds to the real life phenomenon.

Guideline-based expert evaluation [Gabb99], when task are described that is later evaluated using task analysis. In late development phases a heuristic evaluation is performed based on guidelines and expert evaluation on prototypes. This step is really important before going into implementation. Finally, expert guidelines are used to evaluate the resulting 3DUI.

In Bowman, Kruijff, & Laviola [Bowm04], User task (task properties) scenarios are used to validate the usability of the task models. The possible use of IT is evaluated based on taxonomy, metaphor-based [Poup97] or IT-based, decomposition, an interesting approach that theoretically evaluate composition of exiting and proved IT. The context of use (environment, user, system) is analyzed. Early requirements end with a summative evaluation of IT (similarly to [Hix93] and [Poup97]). The concretization of the task modelling is evaluated on prototypes of low-fidelity (similarly to [Hix93]) and IT using Wizard of OZ. The Final result uses a testbed evaluation, including summative evaluation (similarly as [Poup97]).

Finally, there are some other methodologies that just evaluate the final 3DUI and not intermediary steps, for instance, evaluation of the 3DUI with end-users and experts [Cele01] [Neal01]; presence questionnaires [Witm98]; simulator sickness questionnaire [Kenn93]. Similarly, other methodologies just evaluate early stages of the development, for instance, Task analysis [Hack98]; multidimensional design spaces [Card90].

In this review of the literature (Table 2-6), evaluation method used in each development step was analyzed. Taking into account the development step where such evaluation is performed considering the Cameleon Reference Framework as common ground. A dotted line (----) means that the development step is not supported by the methodology. The letters at the beginning of each paragraph denotes whether the method is respectively: automatic (A), manual (M) or both (B).

2.6 Conclusion

In this section, we have selected, analyzed and gathered 3DUI development approaches covering the 3 main axes of this thesis: models, method and language, but also identifying potential software toolkits to support the development of 3DUIs. The first section of this chapter contributes to identify different tools to develop 3DUI from which ideas for the presentation of 3DUI are obtained.

The second part of this chapter helps us to identify the different methods. All the literature review helps us to identify shortcomings on exiting works from which we derive a set of requirements for our new methodology.

On the software toolkit support the *programming approach* allows the direct implementation of the 3DUI. This is mostly the solution preferred by domain experts which are capable to use the different APIs from the different programming languages, to optimize the rendering and performance it is better to work at the lowest level as possible. Programming and maintaining 3DUIs without any method could be not as simple as it gives no guarantee for regularity. There are no evaluation criteria to consider unless the final result is achieved. The tendency is on the “rush to code” approach without any structure favours a “trial and error” method. The result of such a work will highly depend on contingency factors such as the developer's experience or the development context [Limb04c]. Finally, the communication channel between developers and the final users does not exist. So, the programming approach should be taken as it is. Programming a 3DUI is not engineering it [Limb04c]. A *toolkit approach* allows a straightforward implementation of a final interface once modelled in a tool. Using a predefined set of objects that help developers in their programming task, the toolkit approach offers the best solution to draw 3DUIs elements with an immediate feedback. The toolkits reviewed, are very similar in their basic capabilities. Some differences exist on the target language, the model behind, when it exists and the openness (open source projects, software vendors). This section was very useful for this dissertation as it served for selecting the toolkit for designing our 3DUIs.

On the transformational software support our findings showed that there is no best but just depending on the problem you need to select the appropriate one. This information can be relevant to HCI community while trying to tackle the mapping problem for UI development. The comparative analysis contribute to the proper selection of the transformation engine used in this work

On the User Interface Description Languages we also conducted a comparative analysis on existing works. Details on this comparison can be found in the model based incubator group [W3C09] and [Guer09b] where this work has been reported. This comparison can be used to understand and compare the components of different UIDLs in a systematic way –their strengths, limitations, and appropriateness for use. There is currently such a large number of UIDLs available that choosing among them can be time consuming and difficult to do, this comparison can assist UI designers in choosing a language suited to their purposes. This choice is more lead by the goals to be pursued if one decides to adopt one of these UIDLs rather than only the different criteria that have been compared.

On the existing methods there is a plethora of methods and we might have skipped some significant work in this area. However, our goal is not to be exhaustive but to show exiting work. Most approaches cover aspects such as task model-

ling, dialog modelling and implementation aspects. While some knowledge in explicit of existing method most of the time it is hard to find the abstracted models, the transformational approach along with its rules. The following subsections indicate the set of shortcomings identified in the literature reviewed and the requirements established based on the shortcomings. *Shortcomings* are outlined from concerns. A shortcoming is a normative assessment that is made regarding a property of compared transformational methodologies. A shortcoming is normative in the sense that it positions the state of the art with respect to ideal properties identified in the software engineering literature. A set of *requirements* for a solution to overcome the above mentioned shortcomings is identified. The internal validity of the solution proposed in this dissertation will be assessed with respect to this set of requirements.

On the 3DUI Evaluation. Most of the methods apply evaluation just at the end of the development process. Some others introduce early evaluation based on task analysis, scenarios, prototyping. Existing approaches for usability evaluation is bound to the budget of the project. It is always better to have some sort of evaluation, even informal, than nothing. Some methodologies rely on expert evaluation, at different steps of their methodologies, which is not easy to find, not just for the cost but also for their availability. Another problem found in this review is the lack of modality independence: the development is often linked to a particular interaction modality since the early development stages are themselves bound to some interaction techniques, referring to the modality (mode=sense + device) of interaction. The most complete set of guidelines for 3DUI development that covers different development steps was probably presented by [Kaur 98]. We will rely on them for the guidelines to be followed for the final user interface.

2.6.1 Shortcomings Identified from the Literature Review

The shortcomings of the 3DUIs methods (Figure 2-1) surveyed in the current chapter (Table 2-1 and Table 2-2) are inferred from the concerns regarding 3DUIs. These shortcomings lead us to conclude that transformational development of user interfaces can be improved along several dimensions:

Shortcoming #1: Limited support for extensions. Some methods have a particular concern for extension possibilities of their underlying ontology.

Shortcoming #2: Limited transformational approach explicitness. Transformations are in most methods hidden to the designer (i.e., built-in), untraceable and, not modifiable. In some environments, though, rules can be parameterized by dialog wizards (CoGenIVE) [DeBo08]. We are not aware of a 3DUI development method that allows the designer to define custom transformation rules.

Shortcoming #3: Limited outputs for single entries. Methods define their development process with one single entry point (i.e., the development process starts from an imposed artefact) and one single exit point (i.e., the artefact resulting from the development cycle is fixed by the method).

Shortcoming #4: Limited support for the complete Life-cycle development process. In order to develop 3DUIs several methods have been introduced [Bowm04, Cele01b, Fenc01, Geig01, Neal01]. They decompose the software life cycle into steps and sub-steps, but these methods rarely provide the design knowledge that should be typically used for achieving each step. In addition, the development life cycle is more focusing directly on the programming issues than on the design and analysis phases. Available tools for 3DUIs are toolkits, interface builders, rendering engines, etc. When there is such a development life cycle defined, it is typically structured into the following set of activities:

- The **conceptual phase** is characterized by the identification of the content and interaction requests. The meta-author discusses with the interface designer to take advantage of the current interaction technology. The interface designer receives information about the content. The result of this phase is the production of UI schemes (e.g., written sentences, visual schemes on paper) for defining classes of interactive experiences (e.g. class Guided tour). Conceptual schemes are produced both for the final users and the authors. The meta-author has a thorough knowledge of the domain and didactic skills. She communicates with the final user in order to focus on interaction.
- In the **implementation phase**, the UI designer builds the final user interface and the author interface on the basis of the UI schemes. The results of this phase are available as tools for the authors, which can be manipulated without a deep knowledge of computer science world. It is important to note that this implementation phase can be a personalization or a sub-setting of existing tools, rather than a development from scratch.
- In the **content development phase**, authors choose among the available classes of interactive experiences and instantiate the one that fits their particular needs (e.g. guided tour, paths). They take advantage of a number of complementary subjects: editors (writer, 2D graphic artist), 3D modeller, world builder.
- In the **final user interaction phase**, the final user interacts with the contents of the 3d world, composed by the author, through the interface implemented by the interface designer. The final user interaction is monitored in order to improve both the usability of the interface and the effectiveness of content communication.

Shortcoming #5: Limited use of user centred methods. The toolkits follow a content-centric approach, instead of a user-centred approach. Hence, user involvement in the requirements and evaluation are not part of the method.

Shortcoming #6: Limited structured frameworks for the development of 3DUIs. We are not aware of any development framework of 3DUIs that structures the development life cycle in terms of principles to guide designers. Currently, the designer's decisions are not explicitly defined and do not clarify the development of such systems which therefore requires more design workload.

Shortcoming #7: Limited method explicitness. Existing approaches seriously lack explicitness in the way they propose their catalogue of model-to-model and model-to-code transformations both to the designer and to researchers [Limb04b]. The transformation catalogs are implicitly maintained in the head of developers and designers and/or hard-coded in supporting software. Consequently, the transformational processes proposed in the literature consist essentially of black boxes. This lack of explicitness hampers methodological guidance.

Shortcoming #8: Limited modality independence models. All environments deal at the first place with graphical modality but also include models of inputs devices; CoGenIVE [Cupp06] emphasizes the haptic modality but is not limited to it; InTML [Figu04] interaction techniques also consider the input/output channels. Even that exists some modality independent models in the literature, we are not aware of any work that introduced this model for 3DUI models. Due to the continuously increasing number of new interaction devices and as a consequence of interaction modalities that will determine the development of new UIs with new modality capabilities, such a model may avoid their redeployment from scratch [Stan08] and satisfy the *Principle of Separation of Concerns* [Dijk76].

Shortcoming #9: Limited human readability of the ontology. Few methods define in an explicit manner their underlying concepts which are generally bounded to tools or methodological recommendations, thus preventing a designer to grasp the conceptual foundations of a methodology [Limb04c]. Moreover, research teams tend to conduct their researches and developments on their own models which make conceptual consolidation across methods difficult. Cross-method understanding is a tedious and time-consuming activity because it requires understanding the peculiarities of each method and establishing correspondence between them. Thus, communication among researchers becomes complex.

Shortcoming 10: Limited completeness in task modelling. Labels, definitions, goals, and properties used for a task suffer from many drawbacks such as short name, name without action verb or without object (and therefore non-compliant

with the traditional interaction paradigm of action+object), name that is incompatible with its definition, no usage of standard classification.

Shortcoming #11: Limited consistency in task modelling. Labels, definitions, goals, and properties used for a task do not have unique names (e.g., a label or a goal is duplicated), there are some homonyms, and there are some synonyms (e.g., tasks having the same semantics but wearing different names).

Shortcoming #12: Limited correctness in task modelling. Labels, definitions, goals, and properties used for a task violate some of Meyer's seven sins (1985) of specification, i.e., noise, silence, over specification, contradiction, ambiguity, forward reference, and wishful thinking.

Shortcoming #13: Limited support for tool interoperability. Consequently to the lack of explicitness, the exchange of knowledge regarding transformation catalogs can hardly be achieved [Limb04c]. Even when transformation catalogs are made explicit in tools, their heterogeneous formats prevents the reuse of transformations outside the context for which they were designed.

Shortcoming #14: Limited reuse of single language. The reviewed methods are usually restricted to only one programming or mark-up language and do not allow easy porting of code from one platform to another. For instance VRXML is a Markup language for CoGenIVE, VR-Wise has their own language, Contigra extends X3D for their Behavior3D language. They do not rely on existing recommendations from W3C, such as X3D and they built their own language.

2.6.2 Ontological Requirements

We provide a list of requirement we seek to address with this dissertation. Some of these requirements are motivated by the above observations and shortcomings, some are desirable properties found in the literature that apply on any methodology. We want to introduce a 3DUI specification language which:

Requirement #1: The ontology must be extensible (Shortcomings #1, # 8).

States that the ontology structure should allow the extension with new types concepts which include, and are not limited to, new interaction modalities, 3D widgets. This requirement is a principle that we would like to cover, but we are well aware that very complex interactions cannot be supported.

Requirement #2: The ontology must be expressive (Shortcoming #7).

Means that a conceptual framework should provide enough details to address problems that motivated the elicitation of its constituent concepts. In our context models should, at least, provide enough details to allow an implementation of the

system it describes. This essential requirement is not fulfilled by many formal methods, for instance those focusing on verifying state properties of the system that is being built (Motivation: general principle in software engineering).

Requirement #3: The ontology must be human readable (Shortcoming #9).

Means that the provided ontology should be expressed in a format enabling legibility by a human agent more than machine processability [Wing09]. Such efforts are done in InTML, CoGenIVE, VR-WISE and Contigra.

Requirement #4: The ontology should rely on standards (Shortcoming #14).

States that the expression means used to represent our ontology should rely on well accepted standards in the software engineering community. This requirement is desirable but not mandatory, later we will use well accepted notation to express the ontology, which recommendations more than standards.

2.6.3 Methodological Requirements

Requirement #5: Methodological explicitness (Shortcoming #2). States that the constituent steps of our methodology should be defined in a way that facilitates the comprehension of its internal logic and its application.

Requirement #6: Methodological support for multiple solutions (Shortcoming #3). Refers to the option to have more than one output per input model.

Requirement #7: Methodological support for homogeneity (Shortcoming #7). Refers to the property of methodological steps of being defined using a common syntax. All transformation steps should be described in a single formalism that facilitates their understanding and processing.

Requirement #8: Methodological reuse (Shortcoming #4). Refers to the possibility in a methodology to capitalize on the knowledge defined by designers to perform development steps and re-using this knowledge for other times.

Requirement #9: Methodological support for the development life-cycle of 3DUI (Shortcoming #4). States that the methodology should encompass the whole software development life-cycle of 3DUIs.

Requirement #10: The method must be user-centred (Shortcomings #5, #10, #11, #12). States that the method should be explicitly on user needs rather than content modelling. User task modelling must include attributes such as: consistency, correctness and completeness of task models.

Requirement #11: The methodology must be structured and based on principles (Shortcoming #7). Must be structured in a coherent way and driven by principles, not limited to, such as: guidelines, patterns.

Chapter 2. State of the Art

Requirement #12: Support for tool interoperability (Shortcoming #14). Refers to the ability of reusing the output provided by one tool into another. This requirement is motivated by the lack of explicitness of transformations due to their heterogeneous formats that prevents the reuse of transformations outside the context for which they were designed.

Chapter 2. State of the Art

	Models t= task, Do = Domain Di = dialog AUI=abstract presentation CUI=concrete user interface U = user, C = context.	Inter Model Transformation	Platforms	FUI target languages	Avl Mod.
Crazy Eddie	CUI	CUI → FUI	Windows	C++, OGRE	√
Maya	C, A	C → FUI, A → FUI	Mac, Windows, Linux	X3D, VRML, Wave Front	x
Max 3D	C, A	C → FUI, A → FUI	Mac, Windows, Linux	X3D, VRML, Wave Front, C, Alambik, Blitz3D, Anark	x
Anark	C, A, Di	(C, Di) → FUI, (A, Di) → FUI	Windows	Lua script, anark, Maya, Max 3D, Cinema 4D, Light Wave	x
Alice	C, A, Di	(C, Di) → FUI, (A, Di) → FUI	Mac, Windows	Java 3D, HTML	√
Google SketchUp	C, A, Di	(C, Di) → FUI, (A, Di) → FUI	Mac, Windows	Skp, ADT, alibre, allplan, arcGIS, archicad, cadsoft, darkbasic, datacad, live interior 3D,	√
Vivaty Studio	C, A, Di	(C, Di) → FUI, (A, Di) → FUI	Windows	VRML, X3D, 3DStudio, true space, maya	√
Seamless 3D	C, A, Di	(C, Di) → FUI (A, Di) → FUI	Windows	VRML, X3D	√

Table 2-1. Toolkits comparison.

Chapter 2. State of the Art

	Models t= task, Do = Domain Di = dialog AUI=abstract presentation CUI=concrete user interface U = user, C = context.	Inter Model Transformation ↔Bidirectional derivation → Derivation link ≈→ Manual Derivation ≈ ↔Manual Bidirectional der. FUI = Final User Interface	FUI target languages	Avl Mod.	Ext. Mod
VR-Wise	CUI	CUI → FUI	VRML, X3D	√	Orig.
CoGenIVE	T, Di, CUI	(T, Di, CUI) ↔ FUI	C++	√	Orig.
InTML	Di, CUI	T ≈→ (Di, CUI), (Di, CUI) ≈→ FUI	VRML	√	Orig.
Contigra	CUI, Di	(CUI, Di) → FUI	X3D, Behavior3D, Audio3D	√	Orig.
Tres-D	T, U, C, Di, Do	T, Do ≈→ CUI, CUI ≈→ FUI	X3D, VRML	√	Design
Animation	T, CUI, Di	T ≈→ CUI, CUI → FUI	C	x	x
Desktop 3DUIs	T, CUI, Di	T ≈→ CUI, CUI → FUI	X3D, VRML	√	Design
Participative	T, CUI, Di	T ≈→ CUI, CUI ≈→ FUI	Not specified	√	Orig.
Task analysis	T, CUI, Di	T ≈→ CUI	None	√	Orig.

Table 2-2. Model-based methodologies comparison.

	AToM ³	TransformiXML	YATE	AGG
Implementation paradigm	Declarative	Declarative	Imperative	Declarative
Required programming skills	Medium, short learning period	None	High, long learning period	Medium
The model-to-model approach	Graph transformation	Graph transformation	Hybrid	Graph transformation
Code generation	Not supported	Supported	Supported via external tools	Not supported
Pattern matching	Supported	Supported	Supported via external tools	Supported
Rules scheduling organization	Fixed order, no dynamic flow control	Explicit flow control	Explicit flow Control	Fixed order, no dynamic flow control
Rules organization	Distinct rules sets, no inheritance	No limitations	No limitations	Distinct rules sets, no inheritance
Flexibility	Very good	Good	Very poor	Very good
Maintainability	Good	Very poor	Very poor	Good
Completeness	Average	Very Good	Very Good	Average

Table 2-3. Comparison of transformation engines.

Chapter 2. State of the Art

UIL	Models	Methodology	Tools	Supported languages	Supported platforms	Level	Tags	Concepts
DISL	Presentation, dialog and control	Specification of a generic, platform-independent multimodal UI	Rendering engine	VoiceXML, Java MIDP, Java Swing, Visual C++	Mobile and limited devices	Model level	Not specified	Head element, interfaceclasses (structure, style, behaviour), state, generic widgets
GIML	Presentation, dialog, and domain	Specification of a generic interface description.	GTK (Generalized Interface Toolkit)	C++, Java, Perl	Not specified	Meta-model	15 tags	Interface, dialog, widget, objects
RIML	There is no information	Specification of a generic UI description	There is no information	XHTML, XFORMS, XEvents, WML	Smart phone, pda, Mobile, Desktop Pc	Model level	There is no information	Dialog, Adaptation, layout, element
SeescoaXML	Task, Presentation, dialog	Specification of a generic UI description	CCOM (BetaVersion 1.0 2002) PacoSuite MSC Editor	Java AWT, Swing, HTML, java.microedition, applet, VoxML, WML Juggler	Mobile, desktop PC, Palm III	Model level	Not specified	Componem, port, connector, contract, participant, blueprint, instance, scenario, Platform, user, device
SunML	Presentation,	Specification of a generic UI descrip-	SunML Compiler	Java Swing, voiceXML, HTML, UIML,	Desktop Pc, pda	Model level	14 tags	Element, list, link, dialog, interface, generic events, synchronization

Chapter 2. State of the Art

	dialog, domain	tion						
TeresaXML	Presentation, task, dialog	Specification of a generic UI description	CTTE Tool for task Models Teresa	Markup: Digital TV, VoiceXML, XHTML/SVG, X+V Programming: C#	DigitalTV, Mobile, Desktop PC,	Model level	19 tags	Mappings, models, , platform, task, input, output
UIML	Presentation, dialog, domain	Specification of a generic UI description	UIML.net, VoiceXML renderer, WML renderer, VB2UMIL	HTML, Java, C++, VoiceXML, QT, CORBA, and WML	desktop PC, a handheld device, tv, mobile	Model level	50 tags	interconnection of the user interface to business logic, services
WSXL	Presentation, dialog, domain	Specification of a generic UI description	Not specified	HTML	PC, Mobile phone,	Model level	12 tags	CUI=XForms, WSDL, Mapping=XLang Workflow=WSFL, Logic=XML event
XICL	Presentation, dialog,	Specification of a generic UI description	XICL STUDIO	HTML, ECMAScript, CSS e DOM.	desktop PC	Model level	Not specified	Component, structure, script, events, properties, interface
XIML	Presentation, task, dialog, domain	Specification of a generic UI description	XIML Schema	HTML, java swing, WLM	Mobile, desktop PC, PDA	Model level	32 tags	Mappings, models, sub models, elements, attributes and relations between the elements

Table 2-4. Properties Comparison of UIDLs.

Chapter 2. State of the Art

UIDL	Relatedness to UIDL	Standard	Specificity	Publicly available	Level of usage	Weight of the organization behind	Type
DISL	Yes	No	Multimodal UIs for mobile devices	2	3	Paderborn University	Research
RIML	Yes	No	Mobile devices	0	3	Industry: SAP Research, IBM Germany, and Nokia Research Center along with CURE, UbiCall, and Fujitsu Invia	Industry
See-scoaXML	Yes	No	Multiplatform, multidevice, dynamic generation UI	2	3	Expertise Centre for Digital Media Limburgs Universitair Centrum	Research
SunML	Yes	No	Multiplatform	4	3	Rainbow team, Nice University	Research
TeresaXML	Yes	No	Multiplatform, multidevice,	4	3	HCI Group of ISTI-C.N.R.	Research
UIML	Yes	No	Multiplatform	4	3	Harmonia, Virginia Tech Corporate Research (OASIS)	Industry
UsiXML	Yes	No	Multiplatform	5	3	UCL	Research
WSXL	Yes	No	multiplatform, multidevice	4	3	IBM	Industry
XICL	Yes	No	Multiplatform	3	3	Federal University of Rio Grande do Norte, Brazil	Research
XIML	Yes	No	multiplatform, multidevice	4	3	Redwhale Software	Research

Table 2-5. General Features of UIDLs.

Chapter 2. State of the Art

	Task & Concepts	Usability Evaluation Method		
		AUI	CUI	FUI
[Wils02]	(M) Feedback definition (B) Task analysis (M) Interaction techniques selection based on survey of available interactions.	----	----	(M) Fidelity and validity of the virtual environment.
[Cele01]	----	----	----	(M) User and Expert experience evaluation
[Gabb99]	(B) Task analysis (M) Guideline-based expert evaluation	----	(M) Heuristic evaluation based on guidelines expert evaluation on prototypes	(M) Experts guidelines evaluated on the VE
[Nede06]	(M) Usability aspects to be evaluated are defined	----	(M) Design User experiments	(M) User experience evaluation based on questionnaires
[Neal01]	----	----	----	(M) Evaluation on user and expert experience
[Kaur98]	(M) Guideline-based evaluation	----	(M) Guideline-based evaluation	(M) Guideline-based evaluation
[Hack98]	(B) Task analysis	----	----	----
[Witm98]	----	----	----	(M) Presence Questionnaire
[Kenn93]	----	----	----	(M) Simulator sickness questionnaire
[Bowm04]	(M)User task (task properties) scenarios. (M) Interaction Technique evaluation based on taxonomy decomposition (M) Context (environment, user, system) analysis	----	(M)Evaluation on low-fidelity prototyping. (M)Wizard of OZ.	Summative evaluation (M) Testbed evaluation

Chapter 2. State of the Art

	(B) Summative evaluations of IT			
[Hix93]	(B) Summative evaluations of IT	----	(M) Formative evaluation with Prototyping	----
[Card90]	(B) Multidimensional design spaces	----	----	----
[Poup97]	Metaphor-based classifications	----	----	Summative evaluation
[Stee98]	----	----	(M) Guideline-based expert evaluation on prototypes	Cognitive Walkthrough

Table 2-6. Review of existing evaluation methods.

Chapter 3 Ontology of Three-Dimensional User Interfaces

The term "ontology" generates some controversy. It has its history in philosophy, where it refers to the subject of existence. It is also often confused with epistemology, which is about knowledge and knowing. In the context of this research is assumed a set of descriptions of the concepts and relationships within a field of knowledge (3DUIs). This chapter addresses the ontological shortcomings and requirements of Chapter 2 by defining an ontology aimed at describing various concepts relevant to 3DUI development.

Ontology is composed of concepts but from where are they selected? To create the ontology; among other options, two views of the world can be selected whether is objective or subjective. When we refer to concepts that are shared by a community of researchers we refer to a subjective view of the world, as the concepts could differ from one research group to another. The advantage of such a view is that there is no wrong or correct ontology but a suitable one for the purpose that the research is looking for. With the time and its use ontologies could become a standard but then again this doesn't mean that it would be the only way to see this reality it is still a relative view.

Several ontologies have been created to define concepts related to 3DUIs, for instance, there are for interaction techniques [Figu02], behaviour in 3d [Pell05a] and mixed reality applications [Figu06]. In this dissertation, the ontology is more concerned on concepts of 3DUIs, i.e., the representation of 3D widgets, their characteristics, and how from a user task we can derive the 3DUI. This work is incrustrated in the framework of UsiXML [USIX07].

Defining ontologies imply to specify *how* and *what* names are going to be selected to define concepts. The most basic (abstract) foundational concepts and distinctions must be defined and specified. Again subjective views emerge at this step, as there is no way to assure that all the possibilities of concepts in a problem are covered. This work required several iterations which implied the reviewing of lit-

erature in order to enrich as much as possible the ontology, finishing when no changes could be done to it while reviewing more literature. One last step when defining ontologies was the definition of the presentation of the ontology, i.e., the meaning in real world terms of it.

In this chapter we introduce all the abstract concepts of such solutions in ontology that extends the current state of UsiXML ontology [USIX07]. In section 3.1 UML class diagrams for the ontology description along with definitions in natural language, the mathematical structures underlying this ontology, i.e., its abstract syntax described in [Limb04c]. The notion of “directed, identified, labelled typed graph” is introduced, motivated and exposed, finishing with the concrete syntax of the language, which are: the visual (i.e., graphical) and a textual one (i.e., an XML language called UsiXML). The Ontology is compliant to the requirements and is derived based on the requirements established in the state of the art (Figure 3-1). The conceptual content of the extension to this language, which has been created considering just a general perspective of 3DUI, in UML class diagrams is presented. Three essential components are considered: a conceptual content (i.e., abstract concepts), the formal foundations used to represent the ontology (i.e., abstract syntax), and the definition of the appearance of the ontology (i.e., concrete syntax). The structure of this chapter reflects these three aspects.

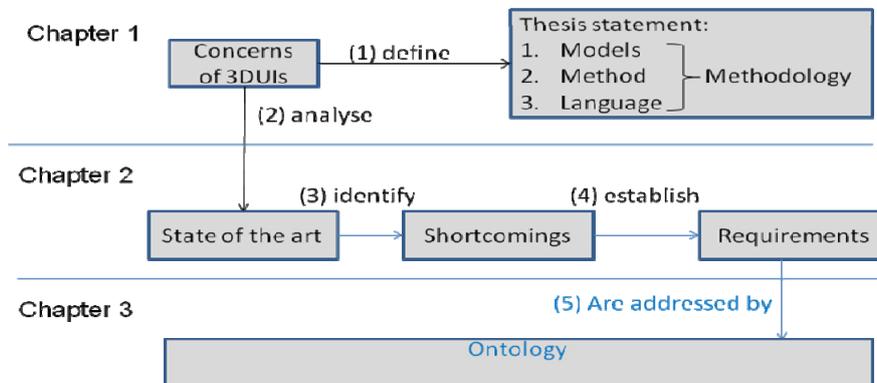


Figure 3-1. General Schema for Ontology derivation.

3.1 Ontology for 3DUI Specification

In this section the abstract concepts of a model-based approach for developing 3DUI is presented. Based on the framework shown in Figure 3-2, the first task is to define the models that we will need. We rely on the chameleon framework [Calv03], see Figure 3-2, with the five models that conforms UsiXML [USIX07]. A complete description of the language can be found in [USIX07] and a brief description of the current models is presented in Appendix B.

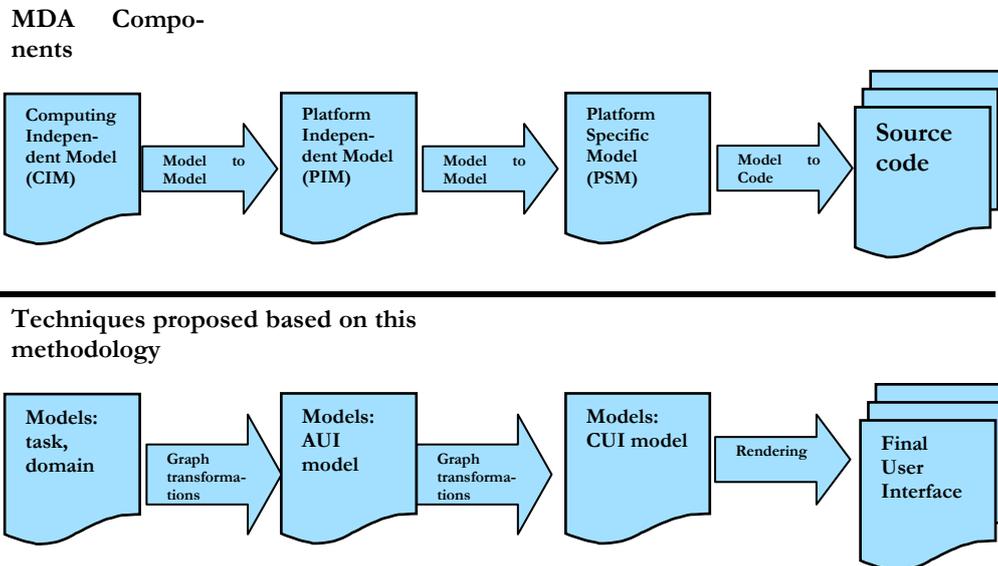


Figure 3-2. The distribution of models according to the MDA classification.

The relevant changes made to the existing ontology are detailed in this section. Particularly, there is a significant contribution to the task model adding a set of enumerated values for action types [Gonz09e]. Being the result of two standardization process [W3C09, Nex09], the domain and AUI models changed. The real impact of those changes is discussed in next chapter where the practical use of such extensions is discussed for task model. In addition, new concepts are needed for the concrete model, mapping model and the context model.

3.2 Task Model

The task model is today at the core of many design activities carried out during the UI development life cycle, such as: user-centred design, task analysis and task modelling, model-driven engineering of UIs, human activity analysis, safety critical systems, and real-time systems. Modelling a task based on well-defined semantics and using a well-understood notation are key aspects, but the many degrees of freedom offered by task modelling should not let us to forget the quality of the resulting task model. The following potential shortcomings were observed:

- *Incompleteness*: labels, definitions, goals, and properties used for a task suffer from many drawbacks such as short name, name without action verb or without object (and therefore non-compliant with the traditional interaction paradigm of action+object), name that is incompatible with its definition, no usage of standard classification.

Chapter 3. Ontology of Three-Dimensional User Interfaces

- *Inconsistency*: labels, definitions, goals, and properties used for a task do not have unique names (e.g., a label or a goal is duplicated), there are some homonyms, and there are some synonyms (e.g., tasks having the same semantics but wearing different names).
- *Incorrectness*: labels, definitions, goals, and properties used for a task violate Meyer's seven sins [Meye85] of specification: noise, silence, over specification, contradiction, ambiguity, forward reference, and wishful thinking.

Not only could those shortcomings be observed during the activity of task modeling itself. But also they could be propagated, if not amplified, throughout the rest of the UI development life cycle since it is effectively based on the task model. The potential damages are even more important if they reach the stage of the FUI. Indeed, several approaches use task models to derive UIs: examples are [Fran93], [Pate99], [Pue97] and [Vand05]. The way of selecting the widgets to be used or the interaction techniques is rather intuitive, while some researchers take this decision on specific attributes of the task model, such as the action type, used by [Vand05], more on this topic is discussed in the next chapter. The UI interaction (UII) is characterized by two elements: (1) the task type (sometimes called action or activity in the literature) and (2) the item manipulated/required in the action. Both attributes are relevant to designing interactive systems.

In 2003, Constantine introduced an abstract set of task types used in common UII [Cons03]. However, an extension to the set of proposed action types is needed, as they do not express a complete abstraction including existing UI action tasks. Due to the fact that most tasks are application dependent it is not feasible to include all tasks types for each existing interaction techniques. So, in order to reduce the design space UI action types from patterns and common UIs are proposed. The benefit of such taxonomy (see Appendix E for more details) is applied when designing the concrete model. At this level the platform where the 3DUI is rendered considers, among other properties, adaptability to different screen sizes, selection of 3D widgets. At this level the nature of the task does not change but interaction techniques are considered and depending on them the set of available 3D widgets vary. According to the current set of action types should consider the abstraction level required for task models that should be independent of modality and platform. For instance, the view task assumes a graphical modality.

The UI interaction is composed of two elements:

- (1) the task type (sometimes called action or activity in the literature)
- (2) the item manipulated/required in the action [Cons03]. Both attributes are relevant to design interactive systems using task models. Indeed, several approaches

Chapter 3. Ontology of Three-Dimensional User Interfaces

use task models to derive UIs [Fran93, Pate99, Puer97, Vand05] where widgets or interaction techniques selection is rather intuitive than systematic. When heuristics are used for the selection researchers based this decision on specific attributes of the task model, such as the task type [Pate99, Vand05].

Naming task types using a restricted set of names has been proposed for different application domains: GUI [Cons03], web interaction [Jans06], input devices [Gree88], haptic interaction [Hutc89, Boda94, Bles90, Limb04b]. Still the names are dependent on the interaction technique to be used and are not generic enough and independent of the implementation (a characteristic that must be accomplished by definition in task modelling). Not everything is lost and few attempts propose canonical description of task types [John95, Fole84], however, they suffer from not being wide enough in order to cope a more general set of task types rather than being too concrete with a limited set of values.

Lenorovitz *et al.* on his review of human computer interaction ended with a list of frequent tasks interactive [John95] separated the task categories into: user interactive actions, user actions and system actions. In this review user actions are more related to cognitive issues. User interactive actions correspond to the tangible manipulation of the system. System actions normally are transparent to the user, they user do not know what is happening at the system level. Constantine proposes a list of canonical Action types and action items, enabling a refined expression of the nature of leaf tasks (sometimes called action tasks or leaf tasks) [Cons03]. This expression qualifies a UI in terms of abstract actions it supports. The list is two-fold: a verb describes the type of activity at hand; an expression designates the type of object on which the action is operated. By combining these two dimensions a derivation of interaction objects supporting a task becomes possible.

Looking at the previous work, from which more than two hundred names were identified, and based on task models found in the literature and done in HCI courses, an agreement was found that: it is not feasible to include all possible names for the task type. Something is needed to reduce the name-space if further transformations are expected from this attribute. From the literature [Gonz09c, Guer08b, Pate02, Pate99, Puer97, Stan05, Boda94], it is clear that the task type is used in further transformations to generate UIs from task models. Then a list of canonical task types is needed. Based on existing list of action types [Bles90, Boda94, Cons03, Fole84, Gree88, Hutc89, Jans06, John95, Limb04b], a comparison of names, meaning and domain applicability was done [Gonz09e] where common task types were grouped (second column in Table 3-1). While looking at the examples and the definition it was clear that they belong to the same category. From there the most abstract name was chosen.

Chapter 3. Ontology of Three-Dimensional User Interfaces

Action Type	Other name options	Definition	Examples
Convey	Communicate, Transmit, call, acknowledge, respond/answer, suggest, direct, instruct, request	The action to exchange information	Show details Switch to summary
Create	Input/Encode/Enter Associate, name, group, introduce, insert, (new), assemble, aggregate, overlay (cover), add	Specifies the creation of an item instance	New customer, blank slide
Delete	Eliminate, Remove/cut, ungroup, disassociate, ungroup	The action of deleting an item	Break connection, Delete file/slide
Duplicate	Copy	Specifies the copy of an item	copy address, duplicate slide
Filter	Segregate, set aside	The action of filtering an item	Filter email, segregate any modification on a data base when backing up
Mediate	Analyze, synthesize, compare, evaluate, decide	The action of intercede task items	Compare products characteristics on a online store
Modify	Change Alter, transform, tuning, and rename, segregate, resize, and collapse/expand	An action of modifying an item	Change shipping address, Tuning volume
Move	Relocate, Hide, show, position, Orient, Path or travel	the action to change the location of an item	Put into address list, move up/down?
Navigation	Go/To	the action to find the way through containers	Navigation bar on a web browser
Perceive	Acquire/detect/search for/scan/extract, identify / discriminate / recognize, Locate, Examine, monitor, scan, detect,	The action of identifying items and/or information from the items	Locate a destination in a map, observe the status bar while installing
Reinitialize	Wipe out, Clear, Erase	The action of cleaning an item	Clear form,
Select/choose	Pick	selection between items	group member picker, object selector
Trigger	Initiate/Start, Play, Search, active, execute, function, record, purchase	Specifies the beginning of an operation	Play audio/video file
Stop	End / finish/exit/suspend/complete /Terminate/Cancel	Specifies the end of an action	Stop searching/playing, cancel register
Toggle	activate/ deactivate, /switch	The existence of two different states of an item	Bold on/off, encrypted mode,

Table 3-1. Canonical list of task types.

In next section the applicability of the action types is shown along with a method to develop UIs from task models [Gree88]. Notice that this set of task types is just a recommendation to be followed but designers could chose any other name for their own purposes but maybe they could found that the name falls into one of our categories. The task items remains without change, see Table 3-2. We change the definition of a collection, to cover a wider set of items and not just elements. The complete description is as follows:

- A *collection* specifies items that are composed of a list of other items. We change this definition by adding the term item instead of element in order to

avoid any misunderstanding. We can have *collection of elements*, for instance, the name, the gender and age of a user. A *collection of containers* is indeed a container with containers, so this attribute is redundant when referring to containers. A *collection of operations* corresponds to a function that call some others to accomplish its task, for instance, a selected object, once the system identified the collision, the representation of a selected object can be composed of: translating the object to the user view point and changing its appearance.

- A *container* when specifies that the item is an aggregation of elements, means, for instance, a group of radio buttons, where the radio buttons are elements or collection of elements. A collection of container it can be boxes in a window.
- An *element* specifies the information from the domain model. For instance the name or the id of client but just one at the time, for composed elements, i.e., classes or tables of a database looks at the *collection* of elements.
- An *Operation* specifies that the item is a function.

task Item	Definition
Collection 	Specifies that the item is composed of a list of task items
Container 	Specifies that the item is an aggregation of elements or collection of elements
Element 	Specifies that the item has a single characteristic
Operation 	Specifies that the item is a function

Table 3-2. Task items.

3.3 Domain Model

The domain meta-model (Figure 3-4) is the result of the NEXOF-RA [Nexo09] effort. Such domain model is independent of the technology or representation mechanisms used for maintaining information in an interactive system.

The *Domain model* is a description of the classes of objects manipulated by a user while interacting with a system. The domain model is composed of *DomainSub-Model*, which is a way to represent a domain model that at the same time can be composed of a different domain sub-model. For instance, Java packages where different submodels can be within a model and they can be associated to different domain facets. A *DomainElement* describes the characteristics of a set of objects sharing a set of common properties. It can be composed of *DomainItem* and *DomainCollections*. A *DomainItem* is an atomic unit of data, i.e. which cannot be decomposed further. The *DomainItem* has an attribute called *itemType* denoting a data

Chapter 3. Ontology of Three-Dimensional User Interfaces

type, such as, but not limited to: Boolean, hour, date, natural, integer, real, character, string. A *DomainCollection* expresses the notion of data collection, i.e. how to collect individual domain items into a structured format; it can be composed of *DomainCollections*. A concrete example of a collection of collections is a table. The *DomainCollection* can be composed of *DomainItems*. A concrete example of a collection of items is a comboBox. The *DomainCollection* attribute called *collectionType* denotes any data structure, such as, but not limited to: set, list, stack, tree, binary tree, shared tree, matrix.

The *DomainRelationship* describes various types of relationships between domain elements. They can be classified in two types: *Constraint* and *FunctionalDependency*. The *Constraint* relationship is self descriptive as it refers to constraints that affect to *DomainElements*. The *Constraint* is described with the *condition* attribute that specifies a condition attached to a constraint, i.e., a rule that must be fulfilled in the specification before or after the application of a *Constraint*. The *Constraint* is specialized in:

- *Existence* denoting the constraint for one *DomainElement* to consider the existence of another *DomainElement*, this is specialized in a *ValueConstraint*, for instance, to pay a flight ticket the user must have inserted their payment method.
- *Relevance* denoting the relevance or importance of a *DomainElement* compared to another *DomainElement*. For instance, in a webmail site the login box is more relevant than the welcome box.
- *CardinalityConstraint* denoting the cardinality constraint from one *DomainElement* to another. For instance, on an ecommerce application the shopping cart has a cardinality constraint to have at least one item when passing to the check out section.
- *ReplicationConstraint* denoting the constraint of replicating one *DomainElement* into another *DomainElement*.

The *FunctionalRelationship* denotes the relationship between two *DomainElements* A and B bounded by a functional dependency if and only if for each value of A, there is at most one value of B. A is called the determinant element, while B is called determined element. This relationship is described with the attribute *semanticFunction* that is the name of a method belonging to the semantic core of the interactive application, such as any method from a class.

3.4 Abstract User Interface Model

The abstract user Interface meta-model (AUI) (Figure 3-5) is the result of the NEXOF-RA [Nexo09] effort. This model describes canonically a user interface in terms of abstract interactors, containers and relationships in a way that is independent from the concrete interactors available on the targets.

In practical terms this means that an AUI model is independent of the target device or modality. This is relevant for 3DUIs development because at this level there is a way to describe the solution without being attached to the Interaction Technique which normally is modality-dependant. The AUI model is composed of *AbstractInteractionUnits* which are composed by *AuiObjects*, *AbstractBehaviour*, and other *AbstractInteractionUnits*.

The *AuiObject* (Abstract User Interface Object) is the root of the hierarchy of User Interface object. *AuiObjects* are the elements populating the AUI Model. *AuiObjects* are described with a set of attributes: *label* is a human-readable label for the object; *longlabel* is a long description of the object; *shortlabel* is a concise description of the object; *help* is a help string to assist users, *contextCondition* is an expression to be evaluated in order to check if the object is relevant under the current Context of Use or not; and *role*, this attribute allows to assign high level semantics to an *AuiObject*. *AuiObjects* can be of two types: *AuiInteractor* and *AuiContainer*. The *AuiContainer* is a container for grouping elements. It denotes the grouping of tasks that have to be presented together, in the same space (windows or page, for example, in a GUI). The *AuiContainer* could contain *AuiInteractor* elements or another *AuiContainer*. The *AuiContainer* is described with the attribute *isSplittable* indicating whether the container could be split in two or more parts, when presenting it to the user (For instance to allow pagination).

The *AuiInteractors* are individual elements populating an abstract container. It is specialized in *DataInteractor* and *TriggerInteractor*. *AuiInteractors* are described with the attributes: *ref* is a reference to the domain model element; *maxCardinality* represents the maximum cardinality for the *AuiInteractor*, and *minCardinality* represents the minimum cardinality for the *AuiInteractor*. The *DataInteractor* is an aggregation of the different types of elements that interacts with the user to present (*Output*) or obtain data (*Input* and *selection*). The *DateItem* is a concept used for the abstract behaviour model with two aggregation relationships (*systemProvided* and *user-Provided*) that represents the system and user provided data directly linked to the *DataInteractor*. This approach is limited for simple values inputs and multiple or composed values. With this definition there is a clear separation between the data it manipulates (*DateItem*) and the type of interactor (*Output*, *Input*, and *Selection*).

Chapter 3. Ontology of Three-Dimensional User Interfaces

The *Input* represents the input facet of a *DataInteractor*, which indicates that such interactor accepts the input of information from the user. The *Output* represents the facet of a *DataInteractor* aimed to present information to the user. The *Selection* is the representation of the facet of a *DataInteractor* that serves for the purpose of entering or displaying information. The *Selection* is described by the attributes:

- *orderCriteria* declares whether the values are ordered
- *isContinuous* declares whether the values are continuous
- *start* indicates the initial value for the set
- *end* indicates the final value for the set
- *step* indicates the difference between 2 contiguous values of the set
- *isExpandable* indicates if it is possible to expand with new values

The *TriggerInteractor* is an interactor that allows users to give orders to the system or to navigate to the functionalities they may need in a given moment. It can have two different facets: *Command* and *Navigator*. The *Command* represents a facet of a *TriggerInteractor* that serves for the purpose of command the system to perform a computation. The *Navigator* is a *TriggerInteractor* facet aimed to change the current presentation unit, thus allowing users to change to a different set of tasks to be accomplished. The relationship between the *AuiContainer* and the *AuiInteractor* is declared with an *AuiRelationship*. The *AuiRelationship* is an association class aimed to indicate explicitly the exact relationship between an *AuiContainer* and its contained *AuiInteractors*. In this sense an *AuiContainer* might have a *hierarchy* relationship indicating that the contained *AuiInteractors* form a *hierarchy*, which is described with a *hierarchy level*. The *Grouping* relationship declares a relationship between two or more *AuiInteractors* grouped in the same *AuiContainer*. The *AbstractBehaviour* corresponds to an abstract view of the behaviour of an *AbstractInteractionUnit*.

The behaviour is an aggregation of *listeners*. A *listener* is associated to at least one *AuiObject*. The *listener* is seen as an ECA rule (on event if condition then action). A canonical set of *abstractevents* and *abstractactions* is proposed for the *events* and *actions*. The list of *abstracteventtypes* is: *onDataInput*, *onErroneousDataInput*, *onDataOutput*, *onDataSelection*, *onDataUnSelection*, *onCommandTrigger*, *onNavigationTrigger*, *onObjectFocusIn*, *onObjectFocusOut*, *onObjectRelevant*, *onObjectIrrelevant*, and *onModelUpdate*. The list of *abstractactions* is: *modelSearch*, *modelCreate*, *modelRead*, *modelUpdate*, *modelDelete*, *modelInvoke*, *modelReset*, *modelCopy*, *objectActivate*, *objectDeactivate*, *objectSetFocus*, *objectCreate*, *objectDelete*, *objectUpdate*, *listenerCreate*, *listenerDelete*, and *eventDispatch*. Notice that both lists are not restricted to these values and can be expanded to any custom value. The *condition* attribute is just a textual expression for the rule to be evaluated.

Chapter 3. Ontology of Three-Dimensional User Interfaces

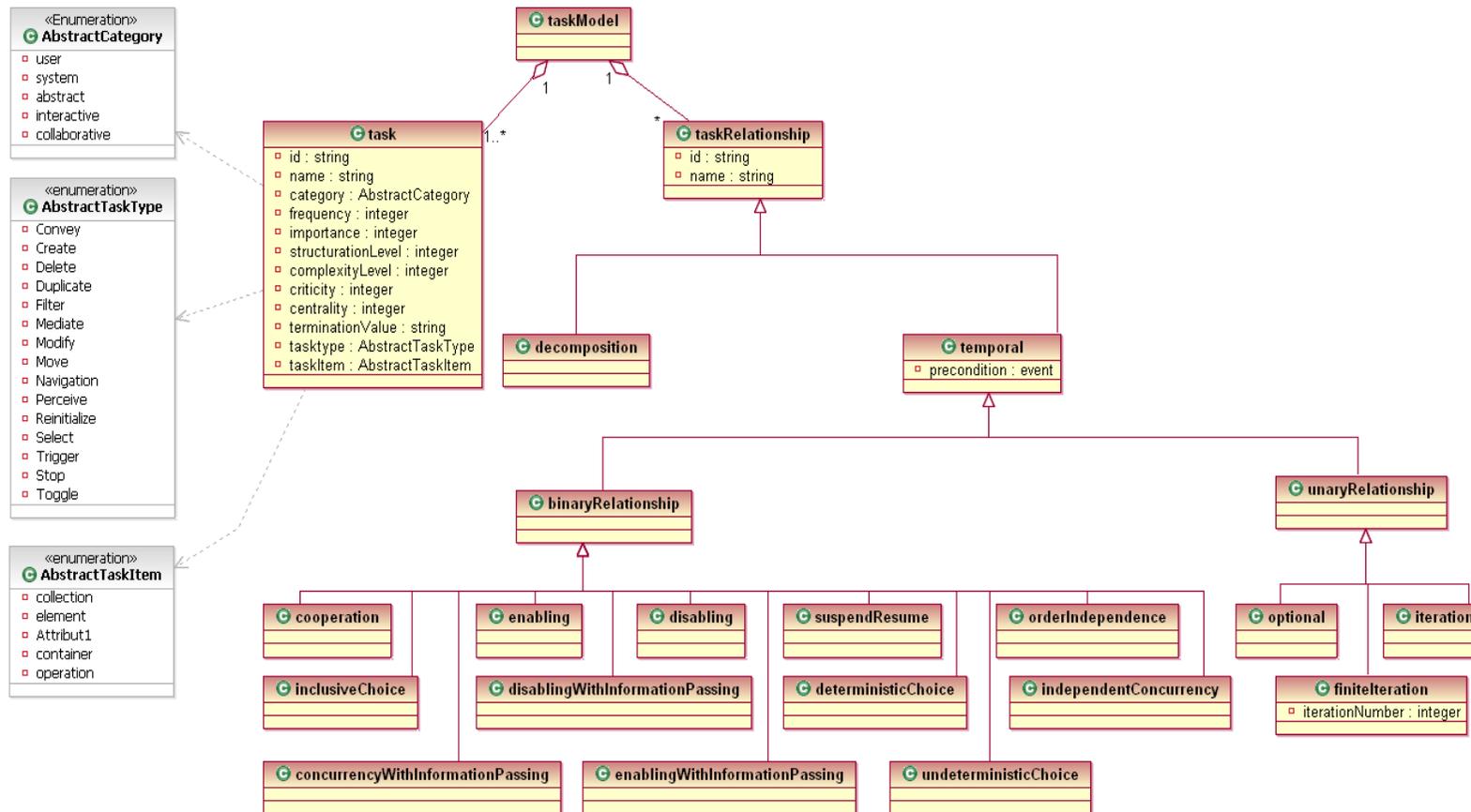


Figure 3-3. Task Meta-Model.

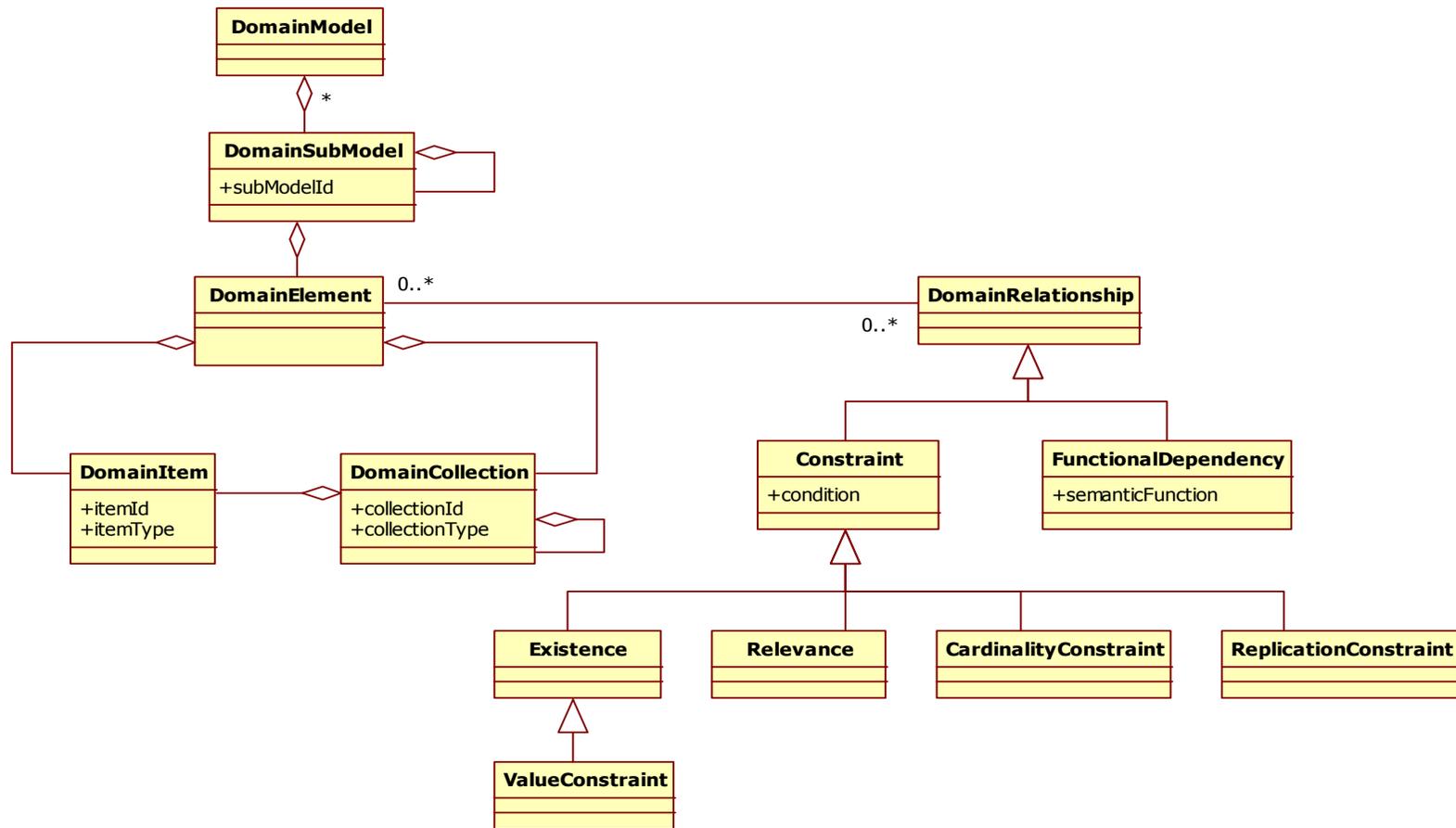


Figure 3-4. Domain meta-model.

Chapter 3. Ontology of Three-Dimensional User Interfaces

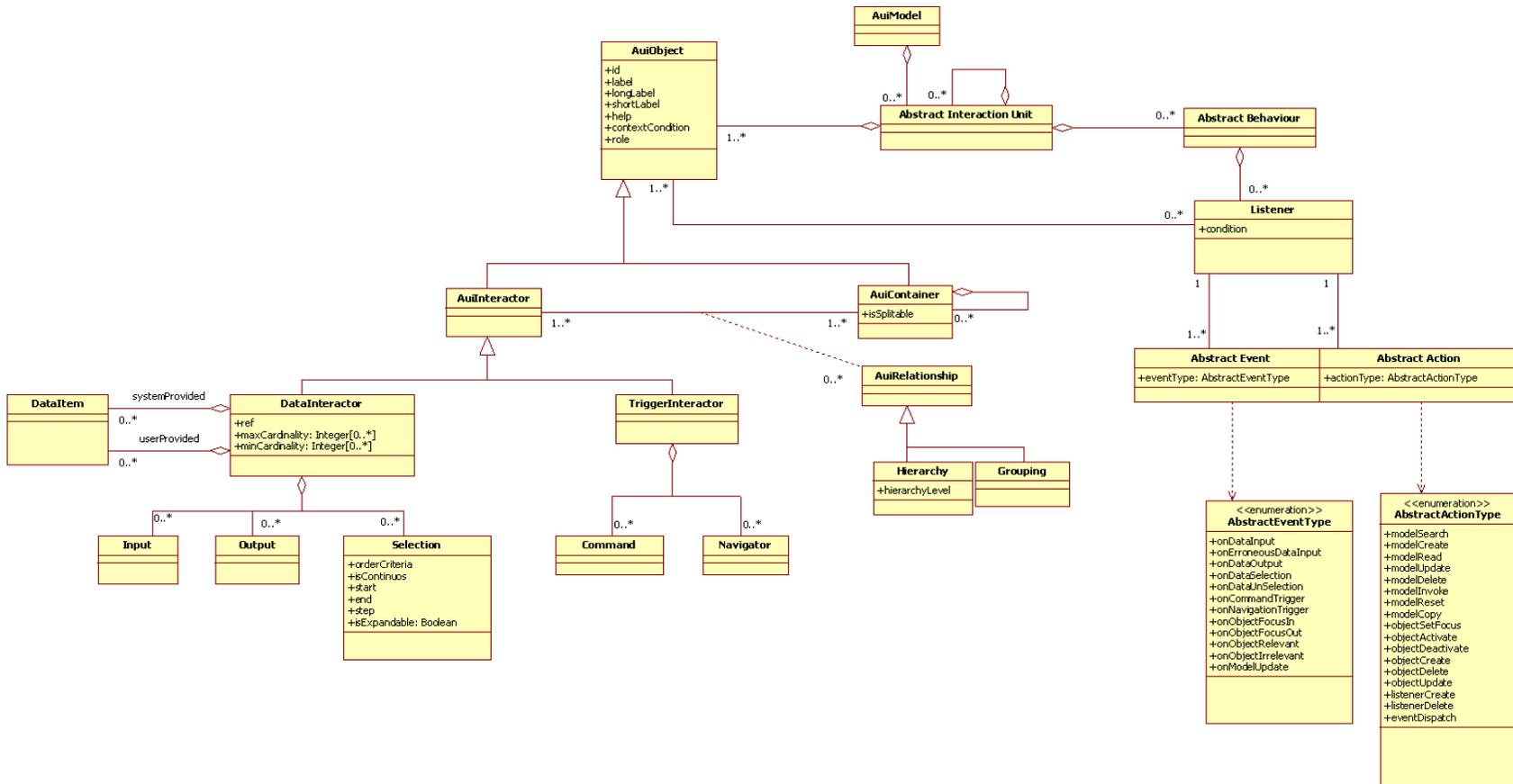


Figure 3-5. Abstract User Interface Meta-Model.

Chapter 3. Ontology of Three-Dimensional User Interfaces

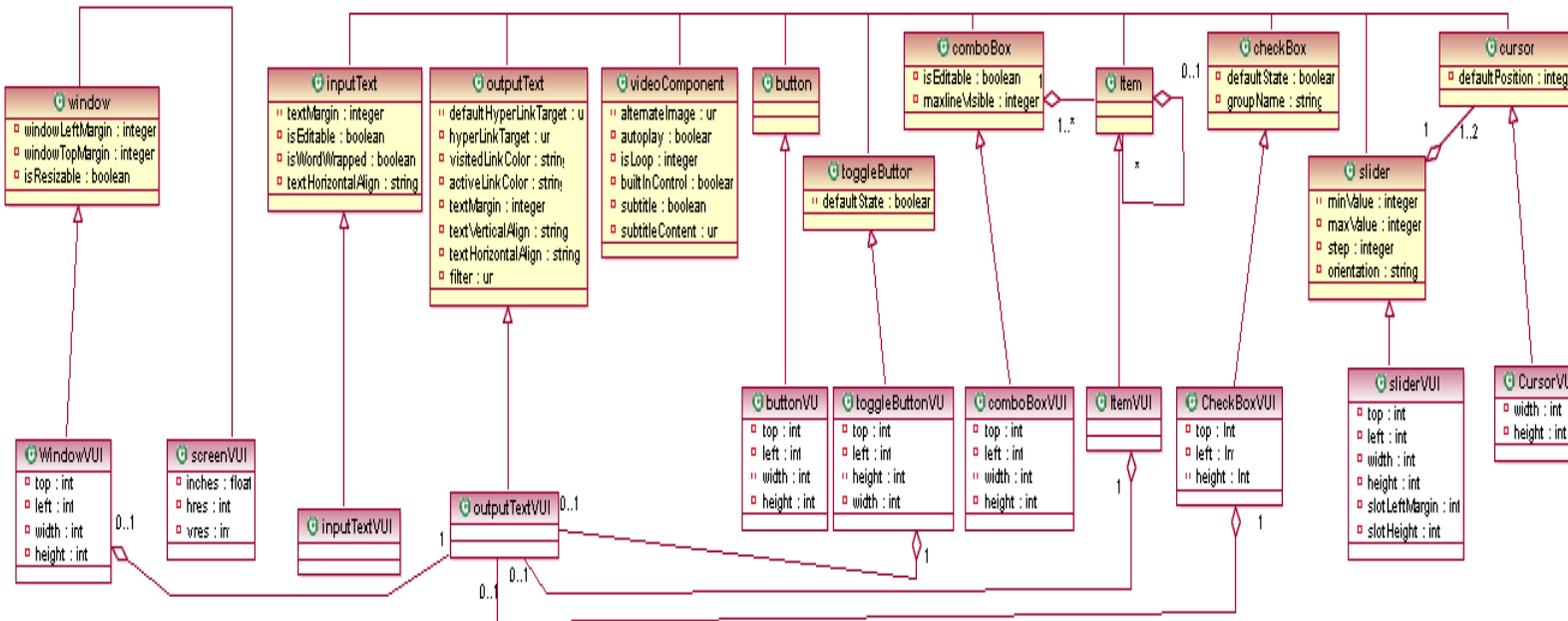


Figure 3-6. Extension of the 2D concrete user interface to support the 3D rendering of 2DUIs.

3.5 Concrete User Interface Model

Much of the time dedicated on developing a virtual world is on modelling objects [Grin96], the content or scene. However, the 3DUI are also part of such development process and represent a lot of effort as well. The concrete user interface model (CUI) describes the 3DUI considering two aspects: graphical representation and haptic interaction. The CUI is composed of two elements: (1) Concrete Interaction Objects (CIO) and (2) Concrete User Interface relationships between them. The CIOs are specialized then into 2D graphical CIO vocal CIO. In this section the extension to 3D graphical CIO is described. Object's appearance and geometry relies on the ISO abstract description of the X3D language [Web304a] for the appearance and the geometry are introduced.

3.5.1 Three-D Rendering of 2DUI Model

A particular case and frequently solution for 3DUIs is the 3D rendering of 2D GUI [Delé08]. The existing model that considers 2DGUI was specialized to consider the attributes specific for their 3D rendering. This work is inspired in the VUIToolkit [Moli05] [Moli08] but it could be applied to any other similar implementation such as Vrixml (section 2.2.2) (Figure 3-8). The VUIToolkit toolkit attributes were used to extend the description of 2D widgets. In Figure 3-6 blue squares around the classes emphasize the VUIToolkit elements. The extension specialize the *CIOs* by adding extra attributes to define their size (height, width) and position (top, left) when it applies [Moli05]. More details of the *CIOs* can be found in the Appendix B.

The major advantages are that 2DUIs have been thoroughly studied and today's users are quite familiar with 2DUI. The problem is that the 2DUIs are not always appropriate for 3D interaction tasks simply because they have not been designed for 3DUIs [Bown04].

3.5.2 Virtual Three-D Graphical User Interface Model

A second approach involves a true 3D presentation of the 3DUI. By true representation is meant going beyond an imitation of their 2D GUI counterpart. In that scenario it was not enough to rely on the existing model as concepts such as appearance, texture, shape, and behaviour were needed. The CUI model was split in two CIOs: 2D based and 3D based. Classes below in Figure 3-6, two groups our 3D Graphical Concrete Interactive Objects (3DGCIO) 3DContainers (3DC) and 3DGraphicalIndividualComponents (3DGIC). The 3DGCIOs has an appearance, behaviour and geometry attached to it. As an infinite variety of graphical

Chapter 3. Ontology of Three-Dimensional User Interfaces

presentations can be proposed for 3DCs and 3DGICs a custom object is proposed to represent the shape of the 3DGCIOS. The attributes inherited from the CIO class were preserved. The resulting extension is based on 3D languages and their corresponding specifications: VRML and X3D languages [Web304a], Maya, Blender (www.blender.org), RawKee (<http://rawkee.sourceforge.net/>), Studierstube and their set of models [Maqu04].

The 3DGCIO is composed of a series of elements (sensors, appearance, grouping components and an abstract object SFNode). The model is depicted in Figure 3-7, attributes for the model are compliant with the abstract definition of the X3D language [Web304a]. The X3D abstract specification focuses on abstract specifications of 3D content in general. In this work we reuse it for 3DUIs in particular but still content could be specified with the SFNode. There are more components for: grouping and event handling (sensors) that can be accessed in [Web304a]. In the remainder of this chapter models are detailed and illustrated in its applicability in the case study chapter. The meta-model for *3DC* and *GCIOs* is detailed in the remaining sections. More objects than those shown has been analyzed but there is no implementation related. Nevertheless, they are included in the following subsections as the design knowledge captured could be useful for the reader.

3D Containers. Ideally, a 3DUI representation is not restricted to a particular image which means that an infinity set of objects, inspired from the real world or not, could be used as containers and interactive objects attached to them. Containers could be 2D rendered in Polygons, irregular or regular, n-sized; 3D renders such as: polyhedrons, prisms, parallelepipeds, pyramids, cones and spheres. Even, custom shapes can be used as container. Consequently, in principle; any combination of shapes can be used. Nevertheless, we consider just those frequently used in the literature and in real-world applications.

Menu. A survey on interaction techniques revealed the existence of Two-dimensional rendering of a menu. A 3D rendering of a 2DUI menu is shown in [Bowm99b]. A menu is a floating bar that could have a background colour that could be visible (Figure 3-9b) or invisible (Figure 3-9a). Properties on the shape such transparency could be modified in order to create such transparent effect on the menu. However, a new attributes is needed to differentiate the type of menu. For instance, when a digital 3D GUI version of the menu is planned, there are several options for menu implementations. In Figure 3-10 a pie menu (a) and a fade-up deering (b), the collapsible menu [Dach01].

Chapter 3. Ontology of Three-Dimensional User Interfaces

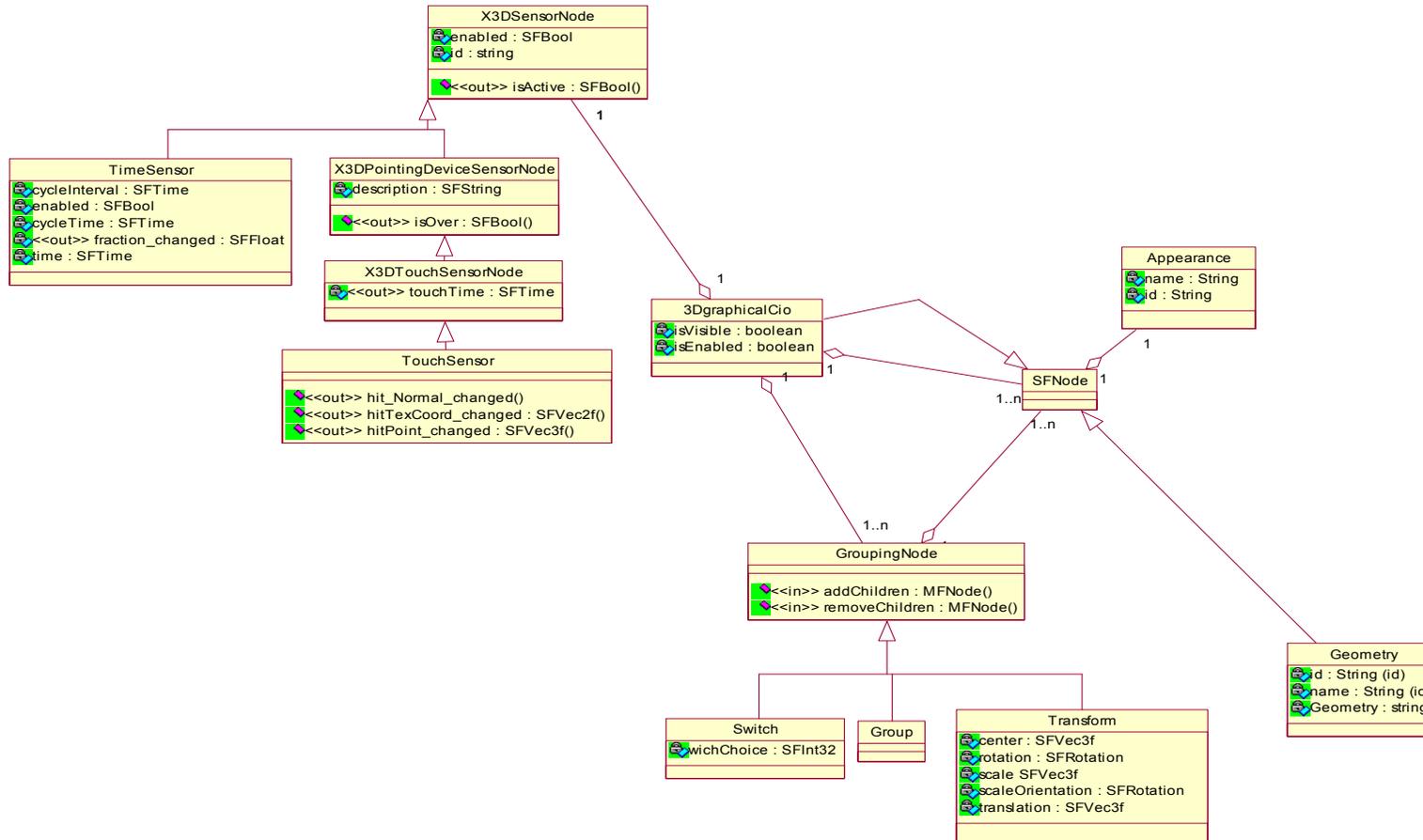


Figure 3-7. 3DGCIO meta-aggregation relationships.

Chapter 3. Ontology of Three-Dimensional User Interfaces

Polyhedron is a three-dimensional shape that is made up of a finite number of polygonal faces. There are different categories of polyhedrons that could be used as 3D Containers for the rest of the *CIOs*, including:

Wall is a rectangular prism, from a rectangle. Text, graphics and many other objects could be attached to a *wall*; also different colours could be assigned to the wall to offer some syntactical related information. Using depth attributes on the wall shape it is possible to build containers such as the one depicted in Figure 3-11. The values of width, height and deep offer the possibility to create different objects, such as cubes. Also different effects could be possible to add to each side of the wall, such as colour, transparency.

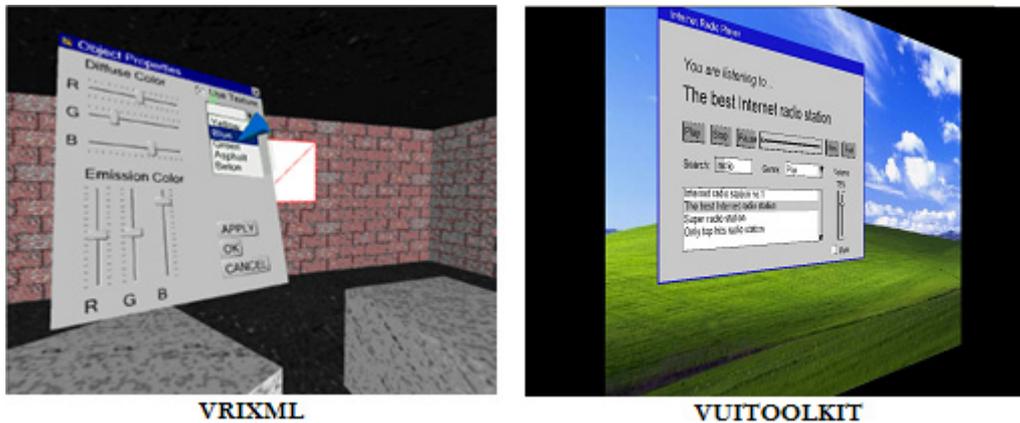


Figure 3-8. Three-D rendering of 2DUIs.

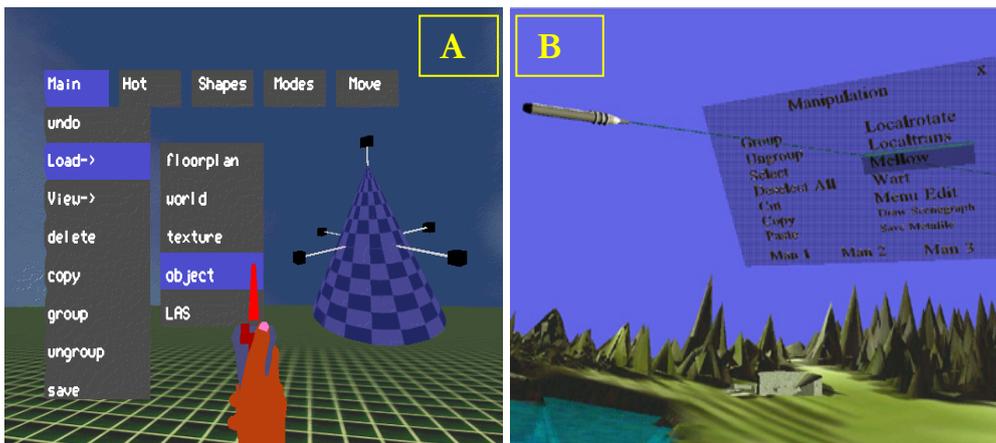


Figure 3-9. 2D Menus [Bowm99b].

Chapter 3. Ontology of Three-Dimensional User Interfaces

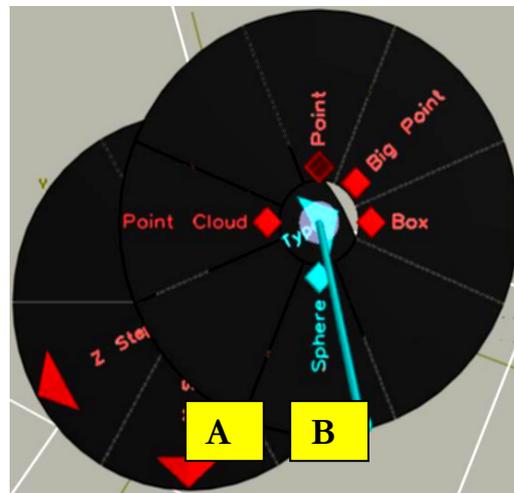


Figure 3-10. Digital version of the Menu [Bowm99b].

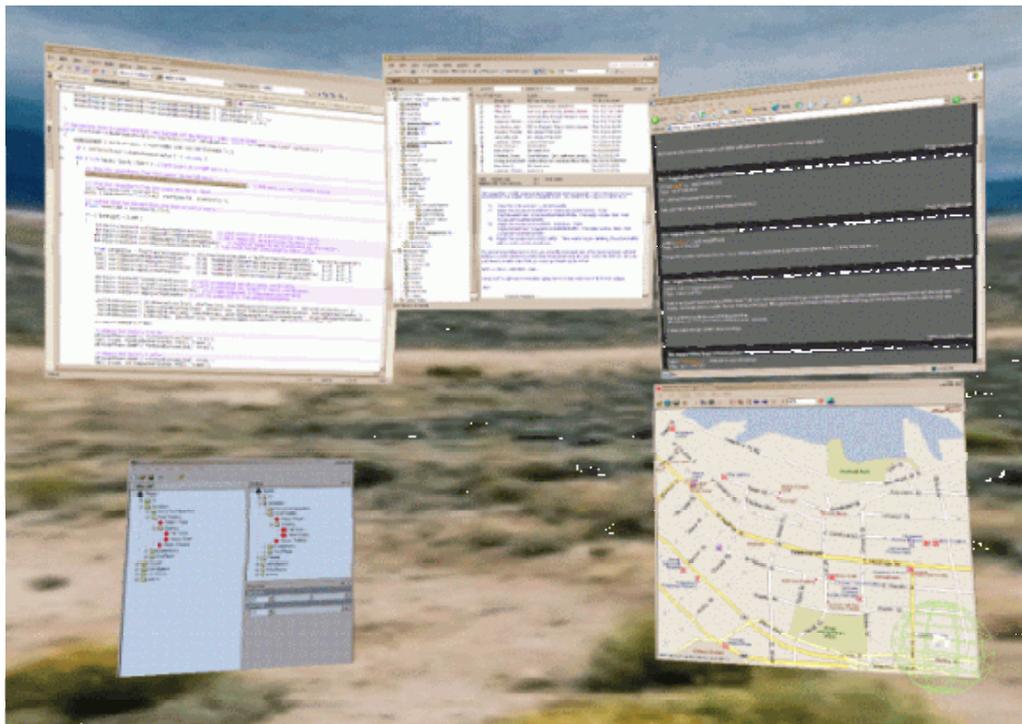


Figure 3-11. Scene with several planes that contains the windows of a Windows system.

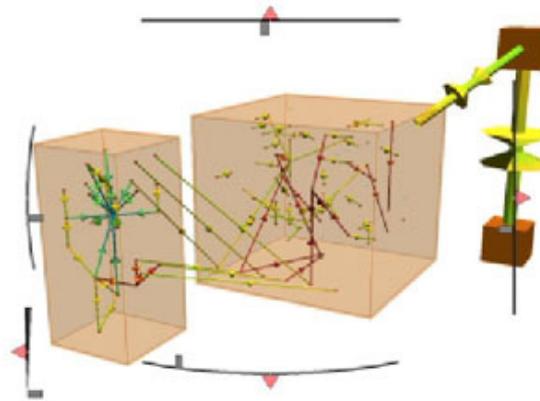


Figure 3-12. Wall Container with transparency [Park98].

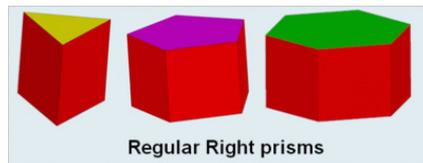


Figure 3-13. Regular Prisms [Wiki05].

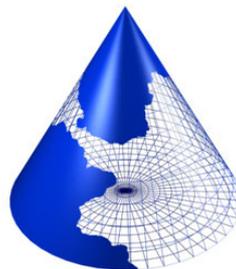


Figure 3-14. Pyramid.



Figure 3-15. Sphere Container [Fair93].

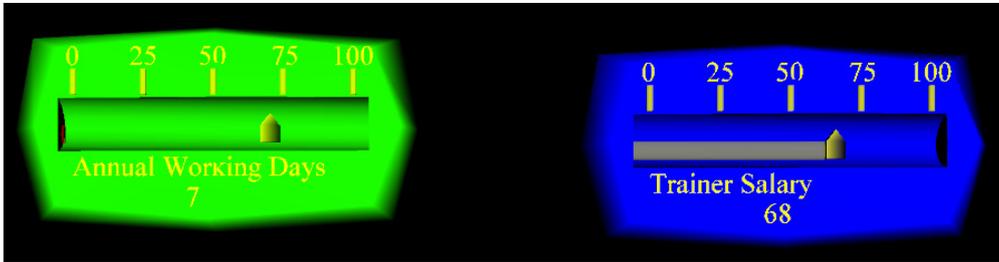


Figure 3-16. Examples of 3D sliders.

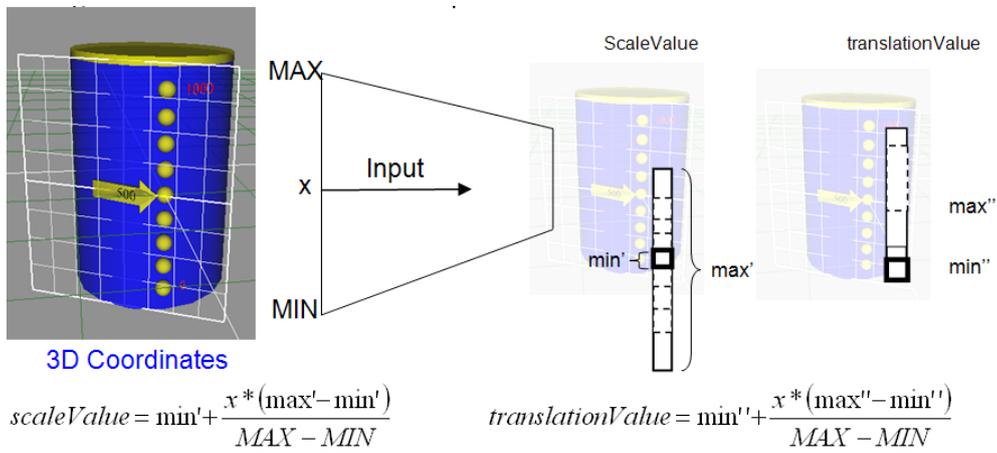


Figure 3-17. Computing the current value on a 3D slider.

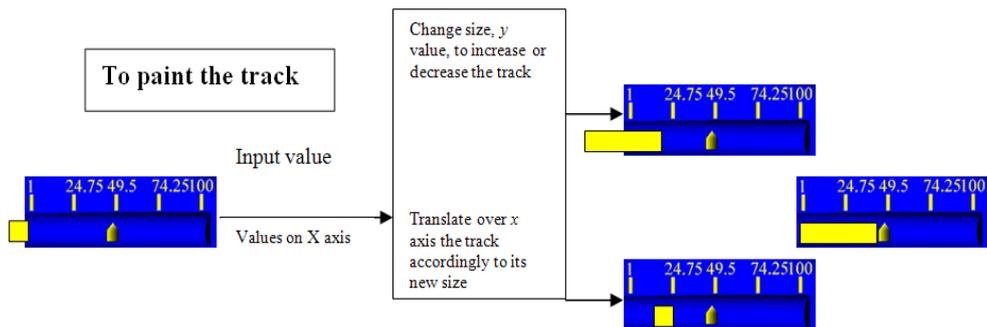


Figure 3-18. How to paint a 3D slider.

Chapter 3. Ontology of Three-Dimensional User Interfaces

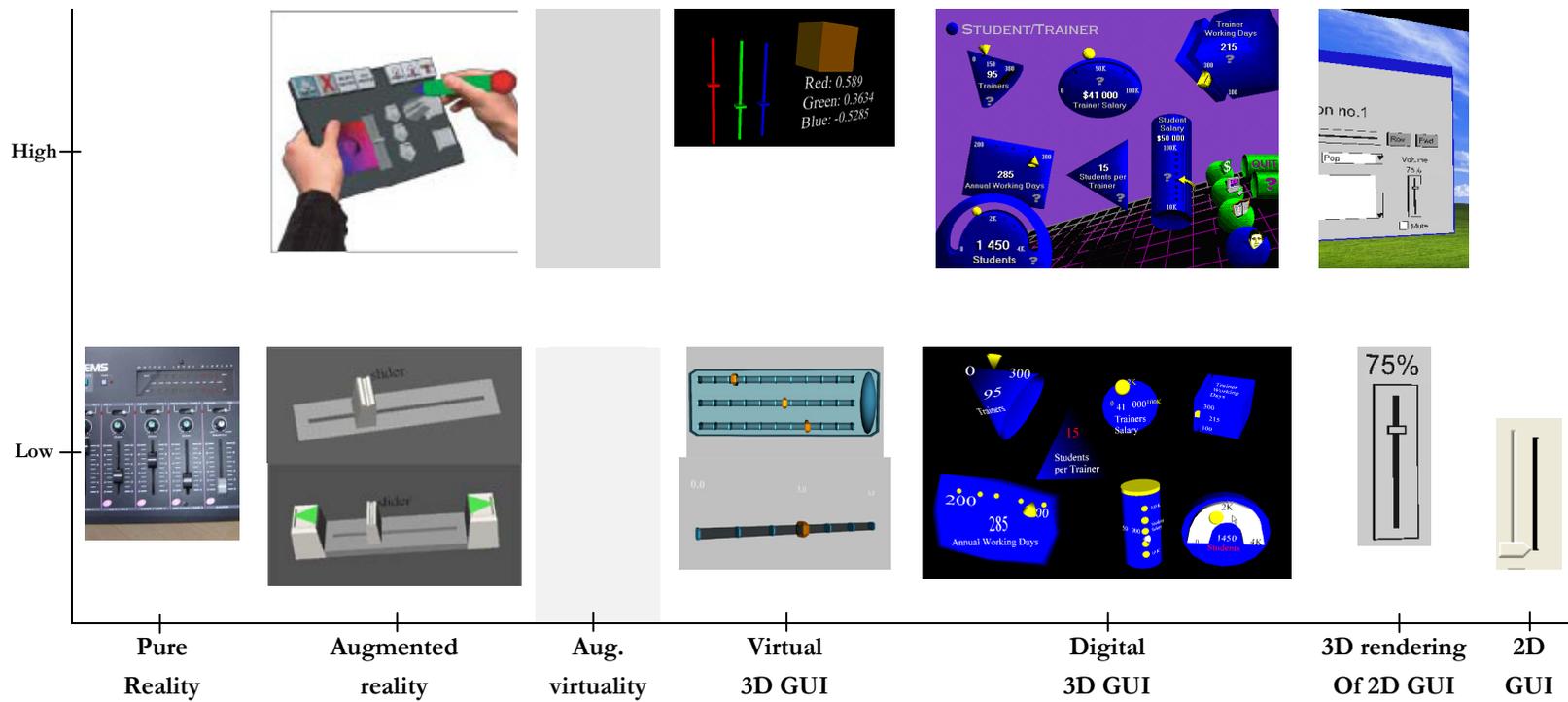


Figure 3-19. Three-Dimensional Sliders Taxonomy.

Prism is a n -sided polyhedron made of an n -sided polygonal base, a translated copy, and n faces joining corresponding sides [Wiki05]. This object container is fully illustrated in a case study (Section 7.1.1). Considering that the prism use a regular polygon as its base the numbers of sizes should be restricted. Even that the size could be n in Table 3-3 is shows that the number of faces should be limited. A software to draw regular polygons was built and the findings are that 17-sides are visualize closed to a circle. A 10-Sized polygon seems to be the most appropriate size. After that, a perspective wall is recommended.

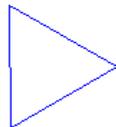
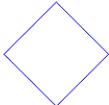
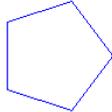
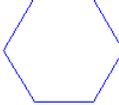
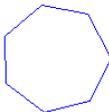
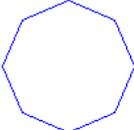
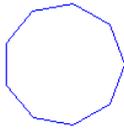
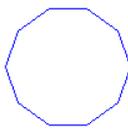
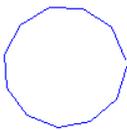
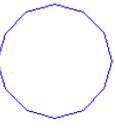
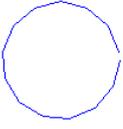
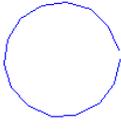
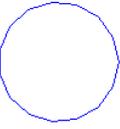
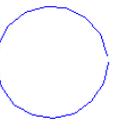
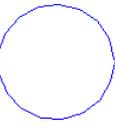
				
3-Sides	4-Sides	5-Sides	6-Sides	7-Sides
				
8-Sides	9-Sides	10-Sides	11-Sides	12-Sides
				
13-Sides	14-Sides	15-Sides	16-Sides	17-Sides

Table 3-3. Polygons approximation to a circle.

A **perspective wall** metaphor [Mack91] could be an option of a newest version illustrated in current 3D Browsers as already depicted in the state of the art in appendix F.

Pyramid is a polyhedron formed by connecting an n -sided polygonal base and a point [Wiki05]. In the literature there was no information where an implementation of such container was used but still it was worse.

Cone is a solid object obtained by rotating a right triangle around one of its two short sides [Wiki05].

Sphere is a perfectly symmetrical geometrical object. In mathematics, the term refers to the surface or boundary of a **ball**, but in non-mathematical usage, the term is used to refer either to a three-dimensional ball or to its surface [Wiki05]. The surface of the sphere could be used to attach objects but also Similar to the wall but instead a sphere could be used as a container (Figure 3-15).

3.5.3 3D Graphical Individual Components

This section describes the different Three-D Graphical Individual Components (TDGIC). The resulting models were abstracted from the literature review and the practical. This section is based on examples developed in the Toolkits reviewed in the state of the art, and the examples shown in the case study, so as some shown inside the taxonomy (see appendix E). The use of the taxonomy provides hints to actual solutions for designers. More important is the fact that there can be more than the examples shown in each level, further investigation can fill the gaps or add more examples to the existing levels.

An illustration of such process is the Three-D Slider. A slider typically refers to the selection of one or two integer value(s) over a set of ordered integers. Contrary to its Two-dimensional counterpart which has just one single graphical representation (Figure 3-19 - 2D GUI), the Three-D slider might have several representations. The abstract attributes of a slider include:

- *minValue*: an integer value to set the lower bound of slider.
- *maxValue*: an integer value to set the upper bound of slider.
- *step*: is an integer value to precise intermediate slider values between min-Value and maxValue.
- *orientation*: a string to define the orientation of the slider, vertical or horizontal.
- *cursorPosition*: a default value for a cursor, that allows to choose a value on a slider.
- *defaultContent*: the text component string that defines the text displayed on the slider.
- *increment*: defines how much the increment or decrement buttons will move the slider.
- *paintTrack*: a Boolean value to define if the track is displayed or not.

Some other attributes associated to the font of the text on the slider and the numbers exist in some 2D implementations but to change the colour of the slider and the tick marks for the steps (Figure 3-16). Extended attributes include:

- *sliderColor*: a three double values set denoting the slider colour.
- *tickColor*: a three double values set denoting the ticks, knob and tickLabels (min, max, intermediate), label colour.
- *trackColor*: a three double values set denoting the track colour.

The range of values of the slider even that rendered in 3D is just in one dimension. Depending on the orientation the corresponding axis is considered, for instance if the slider is vertical then x-axis is used. Another issue to consider is that

while the knob is moved in 3D coordinates the corresponding mapping to an integer value for the real value for the slider is calculated using the formula of Figure 3-17. Another interesting problem was the track line to move along with the knob of the slider (Figure 3-18). It involves scaling and translating the track accordingly to the current value of the knob. Following the same principle used for the knob value (Figure 3-17), 3D virtual coordinate values were mapped for both scale and translation parameters of the track line. Scaling and translating for this particular example is commutative, translation after scaling and scaling after translation has the same result. Both implementations were included in [Gonz09c]. In this section all the TDGIC were not included but they detailed in Appendix C.

3.5.4 Modelling Behaviour

The *behaviour* is the description of an event-response mechanism that results in a system state change. The specification of behaviour may be decomposed into three types of elements: an *event*, a *condition*, and an *action* (ECA rules), in line to the AUI model. [Limb04c] provides a description of the three elements as follows:

- A *condition* is the expression of a state that has to hold true before (pre-condition) or after (post-condition) an *action* is performed. A *condition* may be positive or negative. We express condition as patterns (i.e., a partial description of a state) on the user interface specification itself. Conditions may be composed using traditional logical operator: “AND” indicates a conjunction of conditions; “OR” indicates a disjunction of conditions; “XOR” indicates an exclusive disjunction of conditions; and “IMPLIES” indicates an implication between two conditions.
- An *action* is a process that results in a state change in the system. An action can be of three types: a *method call*, a *transformation system*, or a *transition*.
 - A *method call* is a call to a method that is external to the UI. If a domain model exists, all method calls must reference a method belonging to this model. A method call is normally specified with the name of the method but other referencing techniques are not forbidden, such as scripts. The method call parameters can be specified by making a reference to the value of a property of an object belonging to the CUI.
 - A *transformation system* is the expression of any property change at the UI level. We use a mechanism to specify property changes on the UI. It can be said that a transformation system can be explained as follows: when a pattern is found in CUI specification, changes should occur on the elements matching the pattern. A transformation system might be,

Chapter 3. Ontology of Three-Dimensional User Interfaces

for instance, “when the background colour is white change the foreground colour to blue”.

- A *transition*, also called *navigation*, is a description of a change in the container’s visibility property of a user interface system. A *transition* has a source (a navigation individual component) and a target (generally a container). Transitions may be of different types: open, close, minimize, maximize, tile and restore. Some transition effects may be specified like “box-out”, “box-in”, “fade-out”, “fade-in”.
- At the concrete level an *event* refers to identify not just the movement but also the device. For instance, the mouse device, which are: *move*, *click*, *drag*, and *over*. Events cannot make any reference to coordinates.

Events handling needs a particular treatment for 3DUIs that differs from the previous work in UsiXML. At the AUI level an enumeration class is used to describe a series of abstract events. At the CUI level a 3DUI in this case just handles three types of devices: phantom, mouse, and keyboard. Phantom events are described later (section 3.5.7). For the mouse and keyboard events (Figure 3-20) the sensor mechanism is used [Web304a]. The *X3DSensorNode* abstract node type is the baseline for all sensors. The sensors description, retrieved from [Web304a], is as follows:

- *AudioClip* this abstract node type is used to derive node types that can emit audio data.
- *TimeSensor* nodes generate events as time passes. *TimeSensor* nodes can be used for many purposes including: driving continuous simulations and animations; controlling periodic activities (e.g., one per minute); initiating single occurrence events such as an alarm clock.
- *CylinderSensor* node maps pointer motion (e.g., a mouse or wand) into a rotation on an invisible cylinder that is aligned with the Y-axis of the local coordinate system. The *CylinderSensor* uses the descendent geometry of its parent node to determine whether it is liable to generate events.
- *SphereSensor* node maps pointing device motion into spherical rotation about the origin of the local coordinate system. The *SphereSensor* node uses the descendent geometry of its parent node to determine whether it is liable to generate events.
- *PlaneSensor* node maps pointing device motion into two-dimensional translation in a plane parallel to the Z=0 plane of the local coordinate system. The *PlaneSensor* node uses the descendent geometry of its parent node to determine whether it is liable to generate events.

- *TouchSensor* node tracks the location and state of the pointing device and detects when the user points at geometry contained by the *TouchSensor* node's parent group. A *TouchSensor* node can be enabled or disabled by sending it an enabled event with a value of TRUE or FALSE. If the *TouchSensor* node is disabled, it does not track user input or send events. The field *touchTime* is generated when all three of the following conditions are true:
 - The pointing device was pointing towards the geometry when it was initially activated (*isActive* is TRUE).
 - The pointing device is currently pointing towards the geometry (*isOver* is TRUE).
 - The pointing device is deactivated (*isActive* FALSE event is also generated).
 - More information about this behaviour is described in 20.2 *Concepts*.
- *LoadSensor* monitors the progress and success of downloading URL elements over a network. Only nodes that contain a valid URL field may be specified in the *watchList* field. Multiple nodes may be watched with a single *LoadSensor*.
- *KeySensor* node generates events when the user presses keys on the keyboard. A *KeySensor* node can be enabled or disabled by sending it an enabled event with a value of TRUE or FALSE. If the *KeySensor* node is disabled, it does not track keyboard input or send events. *KeyPress* and *keyRelease* events are generated as keys which produce characters are pressed and released on the keyboard. The value of these events is a string of length one containing the single UTF-8 character associated with the key pressed. The set of UTF-8 characters that can be generated will vary between different keyboards and different implementations. The *actionKeyPress* and *actionKeyRelease* events are generated as 'action' keys are pressed and released on the keyboard. The *shiftKey_depressed*, *controlKey_depressed*, and *altKey_depressed* events are generated as the shift; alt and control keys on the keyboard are pressed and released. Their value is TRUE when the key is pressed and FALSE when the key is released. When a *key* is pressed, the *KeySensor* sends an *isActive* event with value TRUE. Once the key is released, the *KeySensor* sends an *isActive* event with value FALSE.
- *StringSensor* node generates events as the user presses keys on the keyboard. A *StringSensor* node can be enabled or disabled by sending it an enabled event with a value of TRUE or FALSE. If the *StringSensor* node is disabled, it does not track keyboard input or send events. *enteredText* events are generated as keys which produce characters are pressed on the

keyboard. The value of this event is the UTF-8 string entered including the latest character struck. The set of UTF-8 characters that can be generated will vary between different keyboards and different implementations. If a *deletionAllowed* has value TRUE, the previously entered character in the *enteredText* is removed when the browser-recognized value for deleting the preceding character of a string is entered. Typically, this value is defined by the local operating system. If *deletionAllowed* has value FALSE, characters may only be added to the string; deletion of characters shall not be allowed. Should the browser-recognized value for deleting the preceding character is entered, it shall be ignored. The *finalText* event is generated whenever the browser-recognized value for terminating a string is entered. Typically, this value is defined by the local operating system. When this recognition occurs, the *finalText* field generates an event with value equal to that of *enteredText*. After the *finalText* field event has been generated, the *enteredText* field is set to the empty string but no event is generated. When the user begins typing, the *StringSensor* sends an *isActive* event with value TRUE. When the string is terminated, the *StringSensor* sends an *isActive* event with value FALSE.

- *VisibilitySensor* node detects visibility changes of a rectangular box as the user navigates the world. *VisibilitySensor* is typically used to detect when the user can see a specific object or region in the scene in order to activate or deactivate some behaviour or animation. The purpose is often to attract the attention of the user or to improve performance.
- *ProximitySensor* node generates events when the viewer enters, exits, and moves within a region in space (defined by a box). A proximity sensor is enabled or disabled by sending it an *enabled* event with a value of TRUE or FALSE. A disabled sensor does not send events.

Chapter 3. Ontology of Three-Dimensional User Interfaces

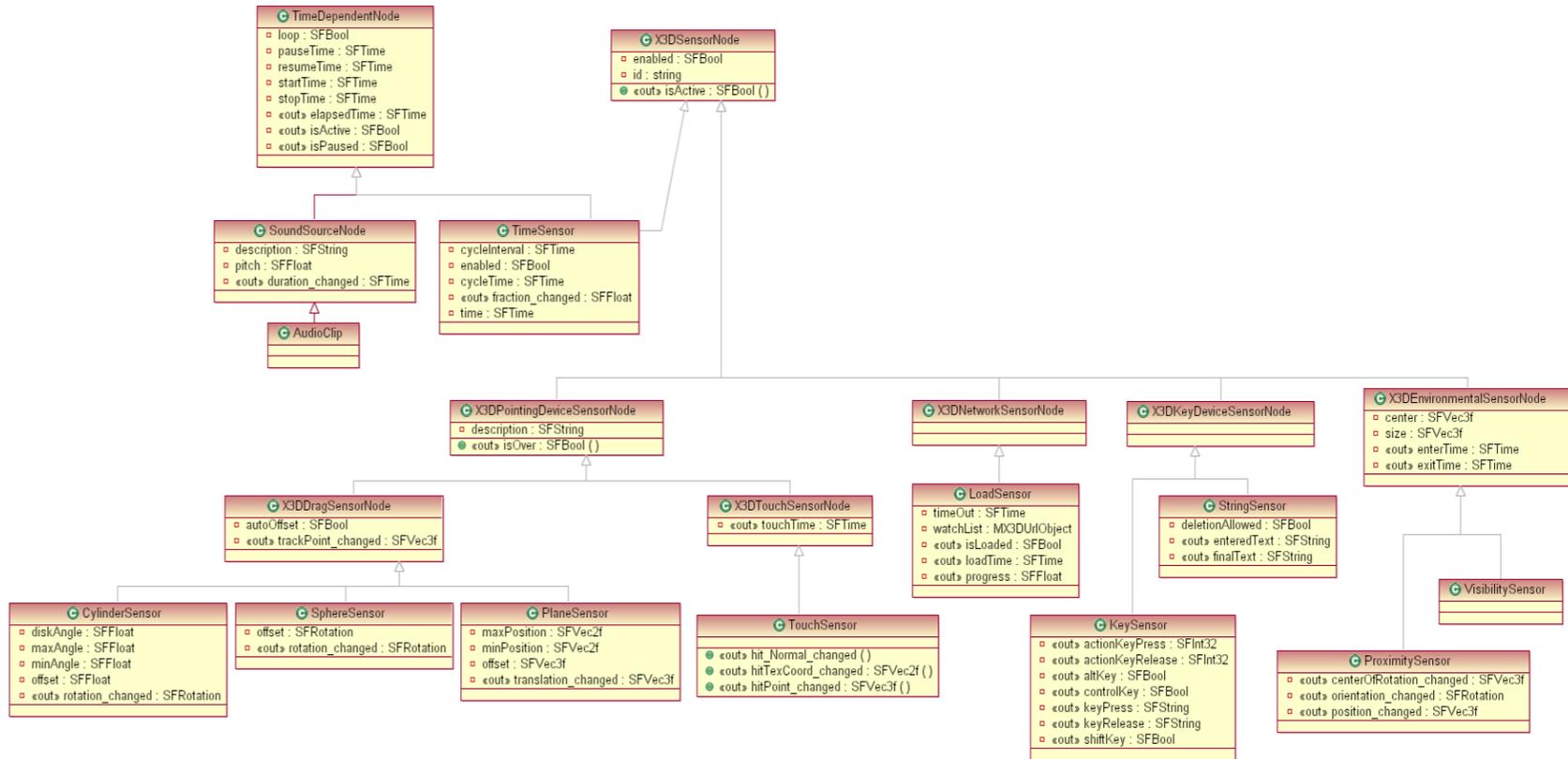


Figure 3-20. Event class for the behaviour model.

Chapter 3. Ontology of Three-Dimensional User Interfaces

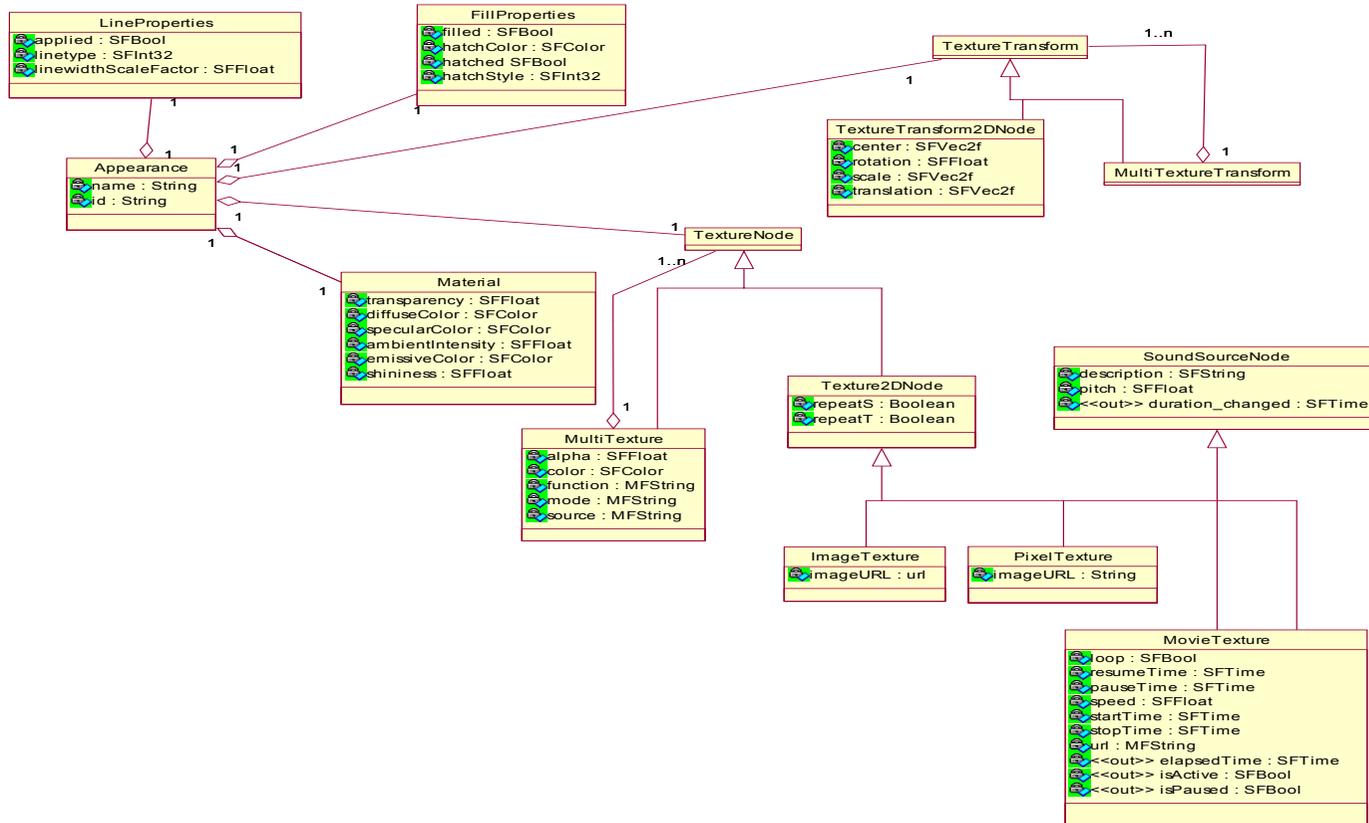


Figure 3-21. Appearance meta-model.

Chapter 3. Ontology of Three-Dimensional User Interfaces

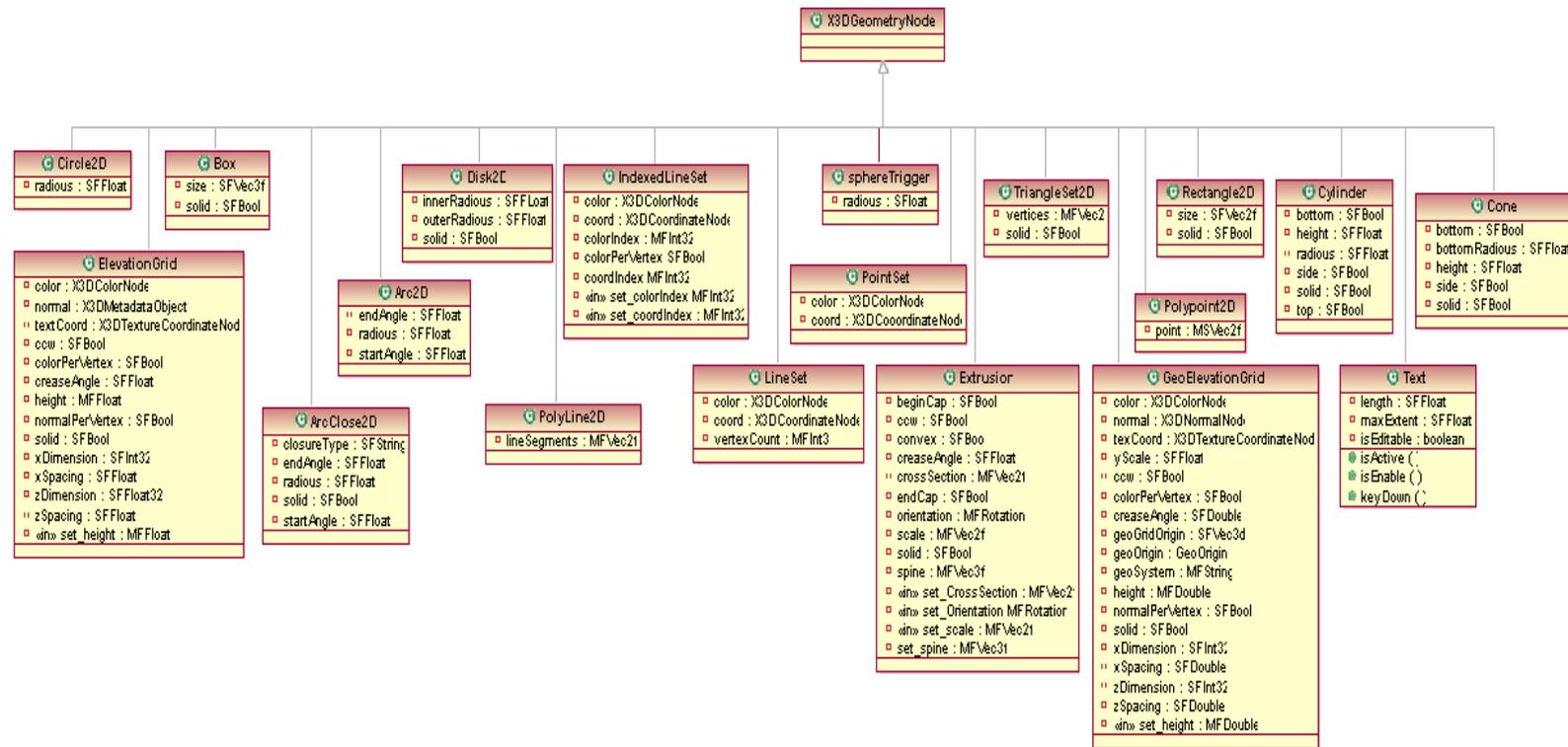


Figure 3-22. Geometry Meta-model.

Chapter 3. Ontology of Three-Dimensional User Interfaces

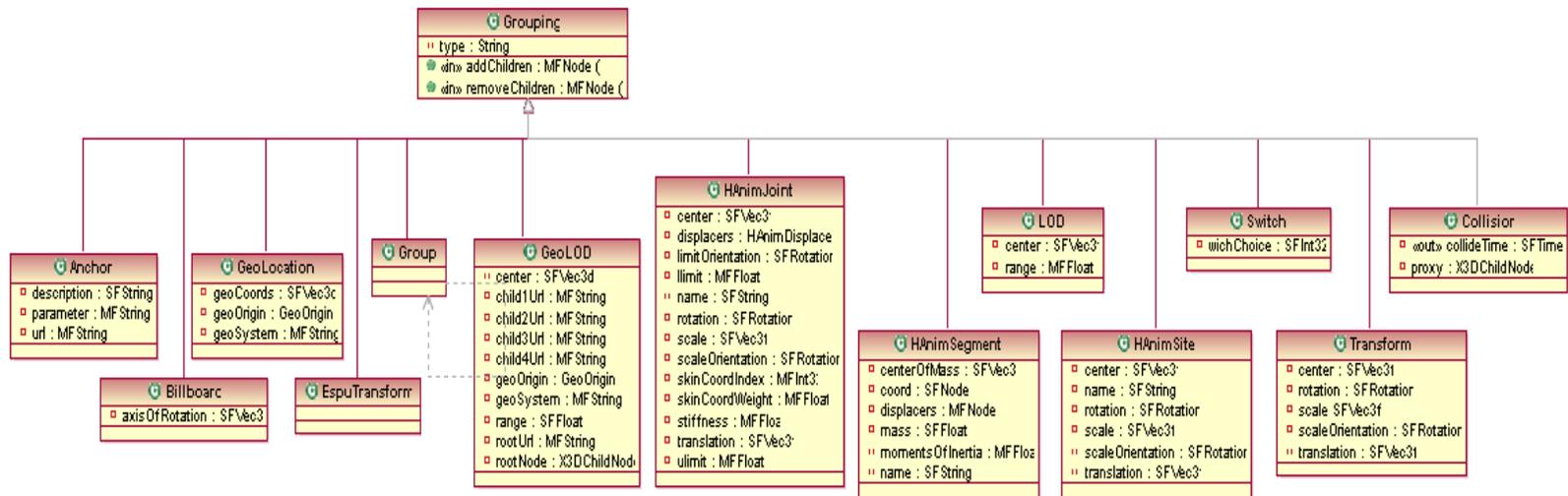


Figure 3-23. Grouping meta-mod

Chapter 3. Ontology of Three-Dimensional User Interfaces

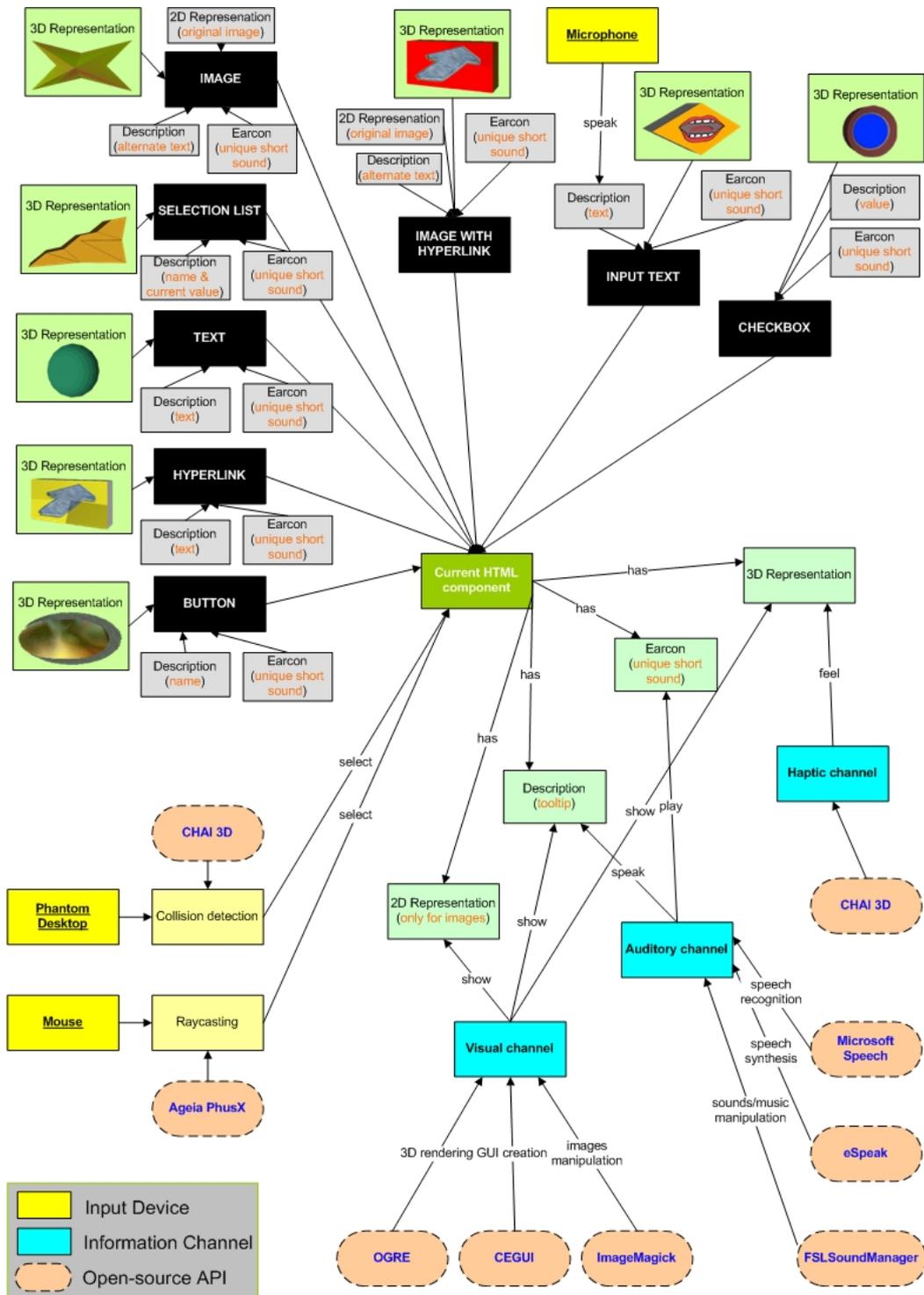


Figure 3-24. Three-dimensional haptic components.

3.5.5 Modelling Appearance

The Appearance node specifies the visual properties of geometry. The value for each of the fields in this node may be NULL. However, if the field is non-NULL, it shall contain one node of the appropriate type. The Appearance element as defined in Figure 3-21 is composed by five elements (Material, Texture, texture transform, fill Properties and line properties). Material properties are related to light and colour whose attributes [Web304a] are:

- *Transparency* defines how clear is the object, with a range of values [0.0(completely opaque), 1.0(completely transparent)].
- *DiffuseColor* indicates the colour reflected by the object, this means, the colour of the object.
- *SpecularColor*. The shiny colour, this means, the shiny spots on the apple.
- *AmbientIntensity*. A double value that indicates the reflection of light from the object.
- *EmissiveColor*. This is useful for shining objects.
- *Shininess*. Combined with the specular colour produce the shine effects, a low value produce soft glows, while higher results in sharper

The LineProperties node specifies additional properties to be applied to all line geometry. The *linetype* and *linewidth* shall only be applied when the applied field has value TRUE. When the value of the applied field is FALSE, a solid line of nominal width shall be produced. The colour of the line is specified by the associated Material node. The FillProperties node specifies additional properties to be applied to all polygonal areas on top of whatever appearance is specified by the other fields of the respective Appearance node. Thus, hatches are applied on top of the already rendered appearance of the node. They are not affected by lighting.

The Texture abstract node type is the base type for all node types which specify sources for texture images. Two are the types of textures that can be applied to objects, multi texture and 2d texture. The 2D texture defines ImageTexture, MovieTexture or PixelTexture, for each of them new attributes are defined. Further information about their properties can be found in [Web304a]. The MultiTexture enables the application of several individual textures to a 3D object to achieve a more complex visual effect. The texture transform specifies how the texture is applied to the object, as two type of texture exists; two texture transformations are required for 2D and for multi texture.

3.5.6 Modelling Geometry

The TDGIC can be composed of several nodes, which can be: predefined TDGICs or any geometry, this class depicts any shape (cube, sphere, and polygons). With these capabilities, any imaginable object for containers and individual components, apart from a set of predefined ones can be obtained. The description of the geometry metamodel, abstracted from [Web304a], includes:

- *Circle2D* node specifies a circle centred at (0,0) in the local 2D coordinate system.
- *ElevationGrid* node specifies a uniform rectangular grid of varying height in the Y=0 plane of the local coordinate system. The geometry is described by a scalar array of height values that specify the height of a surface above each point of the grid.
- *Box* node specifies a rectangular parallelepiped box centred at (0, 0, 0) in the local coordinate system and aligned with the local coordinate axes.
- *ArcClose2D* node specifies a portion of a circle whose centre is at (0,0) and whose angles are measured starting at the positive x-axis and sweeping towards the positive y-axis.
- *Arc2D* node specifies a linear circular arc whose centre is at (0,0) and whose angles are measured starting at the positive x-axis and sweeping towards the positive y-axis. The values of startAngle and endAngle shall be in the range (0, 2p). If startAngle and endAngle have the same value, a circle is specified.
- *Disk2C* node specifies a circular disk which is centred at (0, 0) in the local coordinate system.
- *PolyLine2D* node specifies a series of contiguous line segments in the local 2D coordinate system connecting the specified vertices.
- *IndexedLineSet* node represents a 3D geometry formed by constructing polylines from 3D vertices specified in the coord field. IndexedLineSet uses the indices in its coordIndex field to specify the polylines by connecting vertices from the coord field. An index of "-1" indicates that the current polyline has ended and the next one begins. The last polyline may be (but does not have to be) followed by a "-1". IndexedLineSet is specified in the local coordinate system and is affected by the transformations of its ancestors.
- *LineSet* node represents a 3D geometry formed by constructing polylines from 3D vertices specified in the coord field.

- *Sphere* node specifies a sphere centred at (0, 0, 0) in the local coordinate system.
- *PointSet* node specifies a set of 3D points, in the local coordinate system, with associated colours at each point. The coord field specifies a Coordinate node (or instance of a Coordinate node). The results are undefined if the coord field specifies any other type of node. PointSet uses the coordinates in order. If the coord field is NULL, the point set is considered empty. PointSet nodes are not lit, not texture-mapped, nor do they participate in collision detection. The size of each point is implementation-dependent.
- *Extrusion* node specifies geometric shapes based on a two dimensional cross-section extruded along a three dimensional spine in the local coordinate system. The cross-section can be scaled and rotated at each spine point to produce a wide variety of shapes. An Extrusion node is defined by: a 2D crossSection piecewise linear curve (described as a series of connected vertices); a 3D spine piecewise linear curve (also described as a series of connected vertices); a list of 2D scale parameters; a list of 3D orientation parameters.
- *TriangleSet2D* node specifies a set of triangles in the local 2D coordinate system. The vertices field specifies the triangles to be displayed. The number of vertices provided shall be evenly divisible by three. Excess vertices shall be ignored.
- *GeoElevationGrid* node specifies a uniform grid of elevation values within some spatial reference frame. These are then transparently transformed into a geocentric, curved-earth representation. For example, this would allow a geographer to create a height field where all coordinates are specified in terms of latitude, longitude, and elevation.
- *Polypoint2D* node specifies a set of vertices in the local 2D coordinate system at each of which is displayed a point.
- *Rectangle2D* node specifies a rectangle centred at (0, 0) in the current local 2D coordinate system and aligned with the local coordinate axes.
- *Cylinder* node specifies a capped cylinder centred at (0,0,0) in the local coordinate system and with a central axis oriented along the local Y-axis.
- *Text* node specifies a two-sided, flat text string object positioned in the Z=0 plane of the local coordinate system based on values defined in the fontStyle field. Text nodes may contain multiple text strings specified using the UTF-8 encoding as specified by ISO 10646-1:1993. The text strings are stored in the order in which the text mode characters are to be produced as defined by the parameters in the FontStyle node.

Chapter 3. Ontology of Three-Dimensional User Interfaces

- *Cone* node specifies a cone which is centred in the local coordinate system and whose central axis is aligned with the local Y-axis. The `bottomRadius` field specifies the radius of the cone's base, and the `height` field specifies the height of the cone from the centre of the base to the apex. By default, the cone has a radius of 1.0 at the bottom and a height of 2.0, with its apex at $y = \text{height}/2$ and its bottom at $y = -\text{height}/2$. Both `bottomRadius` and `height` shall be greater than zero.

Grouping nodes, which contain children nodes, are the basis for all aggregation. Several types of grouping elements are defined in [Web304a], including:

- *Group*. A Group node contains children nodes without introducing a new transformation. It is equivalent to a Transform node containing an identity transform.
- *Transform*. The Transform node is a grouping node that defines a coordinate system for its children that is relative to the coordinate systems of its ancestors. The `translation`, `rotation`, `scale`, `scaleOrientation` and `centre` fields define a geometric 3D transformation consisting of (in order): a (possibly) non-uniform scale about an arbitrary point; a rotation about an arbitrary point and axis; a translation.
- *Switch*. The Switch grouping node traverses zero or one of the nodes specified in the `choice` field.
- *Anchor* grouping node retrieves the content of a URL when the user activates (e.g., clicks) some geometry contained within the Anchor node's children. If the URL points to a valid file, that world replaces the world of which the Anchor node is a part (except when the `parameter` field, described below, alters this behaviour). If non- data is retrieved, the browser shall determine how to handle that data; typically, it will be passed to an appropriate non-3D browser.
- *Billboard* is a grouping node which modifies its coordinate system so that the Billboard node's local Z-axis turns to point at the viewer. The Billboard node has children which may be other children nodes. The `axisOfRotation` field specifies which axis to use to perform the rotation. This axis is defined in the local coordinate system.
- *GeoLocation* node provides the ability to georeference any standard model. That is, to take an ordinary model, contained within the `children` field of the node, and to specify its geospatial location. This node is a grouping node that can be thought of as a Transform node. However, the GeoLocation node specifies an absolute location, not a relative one, so content developers should not nest GeoLocation nodes within each other.

- *GeoLOD* node provides a terrain specialized form of the LOD node. It is a grouping node that specifies two different levels of detail for an object using a tree structure, where 0 to 4 children can be specified, and also efficiently manages the loading and unloading of these levels of detail.
- *HAnimJoint*. Each joint in the body is represented by an HAnimJoint node, which is used to define the relationship of each body segment to its immediate parent. An HAnimJoint may only be a child of another HAnimJoint node or a child within the skeleton field in the case of the HAnimJoint used as a humanoid root (i.e., an HAnimJoint may not be a child of an HAnimSegment).
- *HAnimSegment* node is a grouping node that will typically contain either a number of Shape nodes or perhaps Transform nodes that position the body part within its coordinate.
- *Collision* node is a grouping node that specifies the collision detection properties for its children (and their descendants), specifies surrogate objects that replace its children during collision detection, and sends events signalling that a collision has occurred between the avatar and the Collision node's geometry or surrogate. Browsers shall detect geometric collisions between the avatar and the scene's geometry and prevent the avatar from 'entering' the geometry.

3.5.7 Modelling Haptic Interaction

The haptic model extension introduced in this work adds the value of multimodal interaction to 3DUIs, which is one of the main characteristics in 3DUIs. In a multi-sensory design space [Nesb01], many different interaction modalities can be elicited as potential candidates to support a multimodal user interaction [Yu02]. Not all such interaction modalities are equally usable. Although the visual channel probably is the most predominant modality for human-computer interaction in today's computer-based systems, studying when and where an alternative interaction modality may be used instead is still an open and interesting question [Lecu03], whether this is for a so-called normal user [Rams96b] or a person with disabilities [Tzov04]. Equally challenging is the problem of enhancing this visual channel with a second or a third channel in an appropriate way.

Enhancing the 3DUI may consist in enhancing the TDCUIs that compose such a TDCUI with auditory capabilities [Myna94], sonic capabilities [Brew98], tactile feedback [Brew07], or hearing and touching capabilities [Rams96b]. Other interaction modalities could also be considered to enhance TDCUIs, but it is unsure that the resulting combination will be usable enough or superior to the TDCUIs only.

Chapter 3. Ontology of Three-Dimensional User Interfaces

A significant case is when we would like to offer simultaneously visual and haptic interaction: on the one hand the visual channel is preferred by users experiencing no problem in using it; on the other hand, the haptic channel could be used by impaired users, and the combination of both modalities could be offered to those who suffer from sight impairment, but who are not blind: when the visual impairment is important, the tendency would be to rely more on the haptic interaction (haptic dominant interaction) as opposed to the visual interaction when the visual impairment is light, but existing.

Several works provide an audio rendering of web pages to blind users [Avaz06, Laha08, Magn06, OMod97, Rams96A, Rams96b, Sjö02, Tikk06, Wall03, Yu02]. Even the best audio rendering still suffer from some intrinsic limitations such as [Avaz06, Myna94, Sall06]: sequential navigation, long processing time, difficult navigation within a long page or across web pages, audio rendering is independent of any widget and only works when HTML is well-formed [Herm98].

In contrast, haptic interaction displays the abilities to overcome some of these limitations: the user may, in principle, freely navigate within a scene provided that it has been designed to appropriately support haptic interaction (the haptic pointer may asynchronously move from an object to another) no sequence is imposed. Consequently, the time required to switch from one screen object to another object may be reduced at the price of a haptic exploration of the scene. Additionally, via haptic channel the blind users can have a perception of the structure of the virtual environment [Tzov04], in our case the 3D corresponding of a web page, that is very close to the real one (it cannot be exactly the same because 3D rendering puts some limitations in positioning). It is essential not only to give blind people raw information but give them the opportunity to navigate through the internet in a way that makes navigation really interesting.

Experimental studies [Avanz06, Magn06, Rams96a] revealed average search times of items in a scene using the visual channel but there is a need to conduct a similar study to investigate the average reach time of an object using the haptic channel and to identify whether there is some correlation between both search times. It is likely that the exploration time of a “haptic scene” will increase with the augmentation of some parameters like the amount of haptic widgets, their complexity, and the inter-object distance.

Haptic rendering must be easy to customize as the specific needs of each user may differ from one another. The characteristics (shape, effects, surface properties, etc) of each component in a haptic environment have to be changeable. The key of designing a usable UI working for both visual and haptic channels consists in

designing a graphical UI first that is usable enough to appropriately augment it with haptic characteristics that would not affect the initial usability effort.

The result of this analysis ended in **Hapgets** [Kakl08a], a toolkit which satisfies the aforementioned requirements. Its usage is detailed in [Kakl08a, Kakl08b]. In order to support haptic rendering while keeping the classical capabilities of a GUI widgets, we would like to consider the extension of UsiXML (Figure 3-25) involves not just haptic interaction support but also TDCUI.

Rules for haptic rendering. The following rules for haptic rendering of TDCUI followed these rules:

1. In principle, nothing prevents us from haptically augmenting traditional 2D widgets belonging to a native toolkit. This work has already been done for the X Windows window manager [Mill98], which revealed the following conclusion: the haptic augmentation remains only applicable to this computing platform which imposing no particular constraint on the UI to the haptically augmented. Different renderings of a hapget may be offered to convey various meanings to the same object depending on the context. This idea has been extensively illustrated in [Mao00] where the rendering of a task button is varying depending on the frequency of usage (more frequent more rusty), last update (more old more dust)- we would like to introduce the same principle here and to support it through the haptic channel using the surface property.
2. Our human tactile sense is working in 3D since we can sense “tactily” volumes by exploring them on each face. Therefore, it makes sense to presume and exploit our 3D sensing abilities [Magn06].
3. It is possible to include more objects in 3D than in 2D and with a greater degree of complexity [Gonz06a].
4. A 3D exploration of a scene may consider different design options/considerations such as: “putting first-class objects first and delegate second-class objects behind”. “Progressively reveal information through several levels of detail”. These principles could be implemented also in 3D, but with more complexity.
5. 3D spatial relationships may be introduced to convey more complex sets of information that could hardly conveyed otherwise in 2D [Magn06].
6. The shapes of the hapgets included in an interface have to be as simple as possible and without any visual effects. This fact puts many restrictions and limits on the search of the best 3D shape of each hapget. Fortunately, haptic environments offer some other effects that can be used to make the identification of each component easy. Friction, damping and spring are

some characteristic effects that a surface may have. Different combinations between the values of these attributes make the surface of each component unique, even if the same shape for more than one component is chosen. Necessary prerequisite for feeling these effects in a haptic environment is the usage of a haptic device that supports them.

Hapgets: Haptic widgets. The source of inspiration for the widgets was based on symmetry in both graphic and haptic modes, considering the following criteria:

- The *graphical shapes* of the hapget should be different, distinct enough to differentiate the visual appearance of these objects. In particular having two too, practically idem shape should be avoided in order not to confuse the end user. Egg.  and  rounded edge is too close.
- The *haptic parameters* e.g. the hapgets should be different enough to differentiate the tactile sensing of these objects.
- The *volume*, of graphical/haptic, attribute of a hapget should be variable enough for the end user, then the context of use to notice variation of the graph/hap rendering depending on the context. If the variation could not be noticed by the user then the context variation may not be perceived.
- The *graph/hap* parameters should be representative enough of the purpose, the function of the hapget. For instance, a push button should be assigned parameters to preserve the idea of its behaviour, not other behaviour.
- Hapgets should be positioned in space according to a logical order.

The set of widgets for blind user is depicted on the left on Figure 3-24. The shapes of the hapgets included in an interface have to be as simple as possible and without any visual effects. This fact puts many restrictions and limits on the search of the best 3D shape of each hapget. Fortunately, haptic environments offer some other effects that can be used to make the identification of each component easy. *Friction*, *damping* and *spring* are some characteristic effects that a surface may have. Different combinations between the values of these attributes make the surface of each component unique, even if the same shape for more than one component is chosen. Necessary prerequisite for feeling these effects in a haptic environment is the usage of a haptic device that supports them. The haptic device from which we abstracted the model is the Phantom. The effects description is:

- *BuzzEffect*. The effect that vibrates the haptic machine. Its attributes are: Amplitude, duration and frequency.
- *ConstraintEffect*. This effect constraint the haptic machine to the point, line or plane using spring damping system. Its attributes are: damping and springStiffness.

- *inertialEffect*. This effect simulates inertial at the end pint of the haptic machine as if a mass was a task there, using a spring/damping model. Its attributes are: damping, mass and gravity.

The second component relevant to the haptic interaction is the *surface properties*. This model corresponds to the properties of the surface of the 3D components. Their attributes are: static friction, dynamic friction, damping, spring.

3.6 Conclusion

In this chapter, the ontology of concepts required to support model-based development of 3DUIs was introduced. The ontology is compliant to the Cameleon reference framework and to UsiXML. A timeline (Figure 3-26) shows the evolution of the language and the contribution from this thesis.

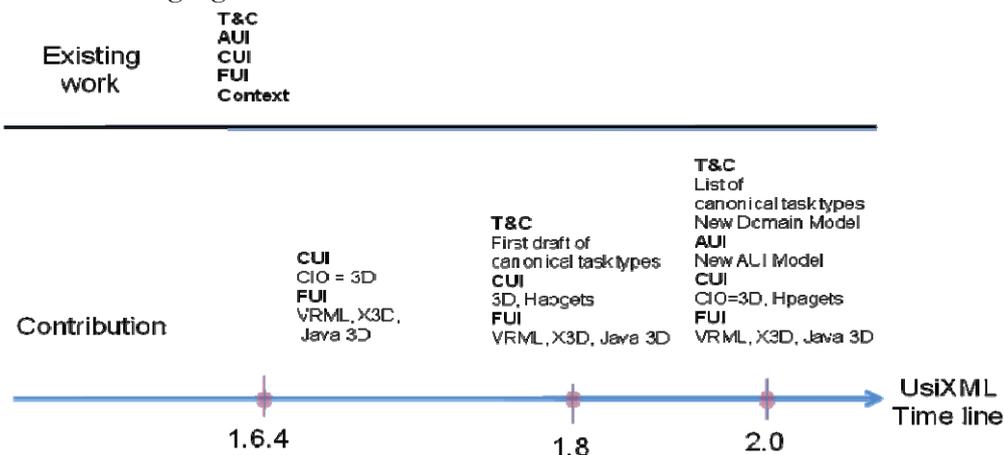


Figure 3-26. Timeline of contributions to UsiXML Models.

In short, the contributions brought by the ontology are the following:

- Task Model, canonical list of action types [Gonz09].
- Domain Model, [Nexo09] and [W3C09].
- Abstract Model, the baseline remains the same but a newer abstraction included in the standardization processes of [Nexo09] and [W3C09].
- Concrete Model, main contribution which is more relevant to 3DUIs, TDCIOs. Extension to model 3D rendering of 2D UIs, graphical (containers and TDCIOs)
- Haptic extension [Kakl08a, Kakl08b].
- Support for modelling behaviour, appearance and geometry of 3DUIs.
- Context model, a new environmental model was introduced accordingly not just to the physical space but to the virtual space also.

Chapter 3. Ontology of Three-Dimensional User Interfaces

- In the next chapter the method that uses the ontology to generate 3DUIs is described. Notice that even a newer version of the Abstract User Interface model and the Domain model has been produced, currently just the semantics of the models are expressed and no syntax. Therefore, hereafter previous UsiXML models, which can be found in Appendix C, were used for the method and the cases studies.

Chapter 4 A Method for Developing 3DUIs in a Principle-based Way

In this chapter the method used in this methodology is presented. The goal is not to come up with yet another Software Development Method but to reuse existing work and structure it accordingly. The result is a method that structures the development life cycle of a 3DUI of an IS in a principle-based way. The method follows an exploratory approach as its goal is to show a variety of possibilities to encourage design. It is said that is structured as it is based on a structured Framework, the Cameleon Framework. Second, a set of instruments (evaluation guidelines, patterns, canonical task types, rules for model to model transformations) guide de use of the method then making it principle-based. The method addresses methodological requirements and is based the Ontology (Figure 4-1).

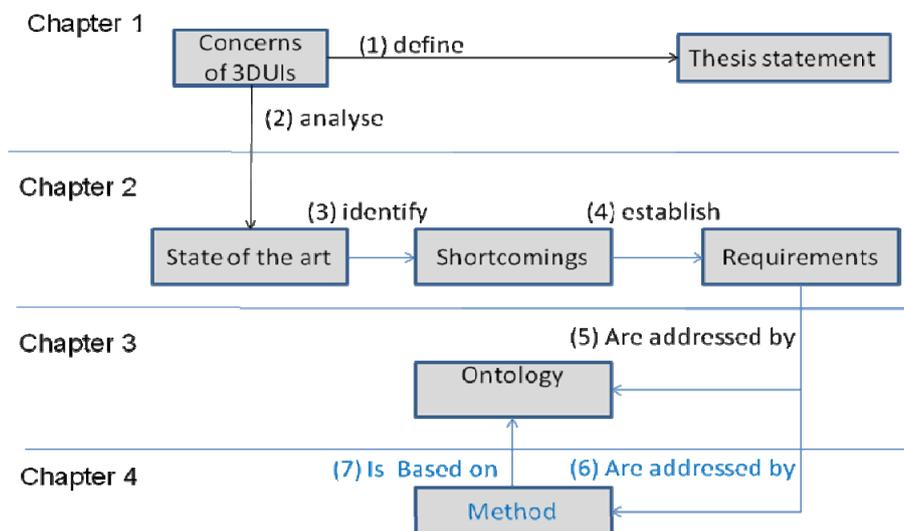


Figure 4-1. General Schema for Method derivation.

By expressing the steps of the method through transformations between models, the method adheres to MDE paradigm where models and transformations are explicitly defined and used. The use of a formal specification technique is extremely valuable, because it provides non-ambiguous, complete and concise ways

of describing the behaviour of the systems. The method uses a set transformation rules to pass from one development step to another. Usability guidelines are also encoded as a subset of transformation rules that are applied at the different steps of the method. These rules can generate a final result automatically, if a software tool does the work, or could be used manually, even using a software tool authors could be asked to make decisions on the guideline whether follow it or not. In order to ensure the three first steps, transformations are encoded as graph transformations performed on the involved models expressed in their graph equivalent. In addition, a graph grammar gathers relevant graph transformations for accomplishing the sub-steps involved in each step. Once a CUI is resulting from these three first steps, it is converted in a development environment for 3DUIs where it can be edited for fine tuning and personalization. From this environment, the UI code is automatically generated.

The remainder of this chapter is structured as follows: the proposed method is discussed including guidelines for each transformational step. This chapter ends with conclusions including a summary the benefits of this method.

4.1 Software Development Life Cycle of 3DUI for Information systems

Web Engineering for 3D Web application is an area in which model-driven software development can be successfully applied in a similar way as for GUIs [More07]. The proposed development method [Gonz06a, Gonz06b, Gonz08a, Gonz08c, Gonz09a, Gonz09c] is structured accordingly to the Cameleon Reference Framework [Calv03].

We conducted a survey on the different evaluation methods for 3DUIs (section 2.5) from which we learnt that evaluation methods can be classified according to three axes: User involvement, this characteristic illustrates methods that require users either beta or experts and those that do not; type of results, this characteristic identifies whether or not a given usability evaluation method produces qualitative or quantitative data; and context of evaluation, this characteristic represents the type of context in which the evaluation is conducted, it can be a generic context of use or simply an application-specific one, for which results remain specific. The characteristics are not mutually exclusive [Bowm02]: as they all form a design space for evaluation design. The present work is situated on different axes, depending on the development step. The evaluation at each step can be performed manually or automatically. In order to provide yet another possibility to evaluate 3DUI, our method could be an option to cope with the budget problem. The method still has some open issues as discussed in [DeBo06], from:

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

- A *technological point of view*: it involves an integration of technologies to support the complete process. A transformation engine to support the transformational approach, high-level editors to support the design of concepts at each development step, a change tracking system (reverse engineering process) to identify changes in dependent models are also beneficial in a mature model-based approach.
- A *methodological point of view*: there are quite some open issues for which the solution is not straightforward.

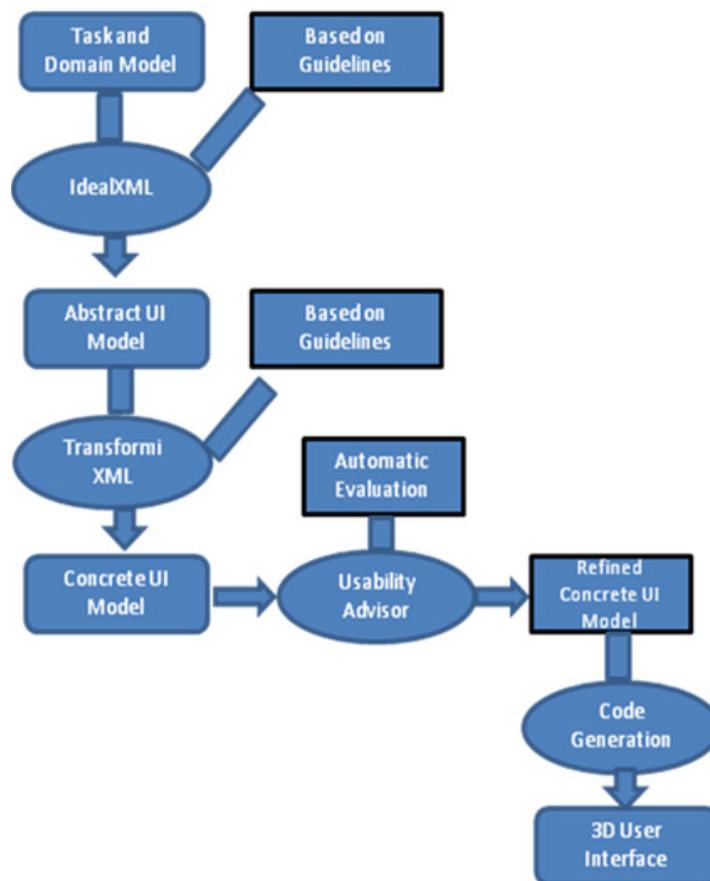


Figure 4-2. Outline of the method for developing 3DUIs.

One more issue refers to methodological aspects addressing guidelines and heuristics that can be manually or automatically incorporated in the development steps in order to contribute on the creation of usable 3DUI. The next subsections discuss each sub-step and how principles can be incorporated to guide designers on the use of best practices when modelling 3DUI at the different abstraction levels proposed in our method.

4.2 Step 1: Task and Domain Modelling

The task model is today at the core of many design activities carried out during the UI development life cycle, such as: user-centred design, task analysis and task modelling, model-driven engineering of UIs, human activity analysis, safety critical systems, and real-time systems. Modelling a task based on well-defined semantics and using a well-understood notation are key aspects, but the many degrees of freedom offered by task modelling should not let us to forget the quality of the resulting task model. Over time, we observed potential shortcomings: limited completeness, structure, correctness when naming tasks. Not only could those shortcomings be observed during the activity of task modelling itself. But also they could be propagated, if not amplified, throughout the rest of the UI development life cycle since it is effectively based on the task model. The potential damages are even more important if they reach the stage of the FUI. Indeed, several approaches [Fran93, Pate99, Puer97, Vand05] use task models to derive UIs and the way of selecting UI widgets or interaction techniques is rather intuitive than systematic.

In order to provide some means to designers for task modelling we propose: 1) to reuse existing successful solutions existing as task patterns and 2) to follow a set of guidelines that might lead to a consistent task model that later can be object of transformations.

4.2.1 Three Dimensional Task Patterns

The work of Alexander on pattern language for architectural designs [Alex77] has been applied in HCI domain. HCI patterns have been used to specify and to capitalize good practices to design graphical UIs in different domains, such as: interactive patterns [Weli00], usability patterns [Mahe01], task patterns [Prib03], games design patterns [Davi04] and design patterns [Gamm95]. The idea of using patterns for task models is not new. The goal is to simplify designer's workload and to have just one single representation of well known tasks (patterns). By using this approach, it can be ensured, in some way that further reifications of the task models to UIs found in the subsequent levels (AUI, CUI, and FUI) are valid.

There has been a lot of work regarding the modelling that captures the essence of successful solutions for 3DUI development, majorly for interaction techniques. Modelling interaction at the task model require a level of abstraction that is independent of the modality of interaction, thus interaction techniques should not be detailed in a task model. Rather, the user task to be performed in the 3DUI. Findings from [Bowm04] showed that interaction in 3DUI can be expressed in four

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

main tasks, the Universal 3D interaction tasks, which are: Navigation, Selection, Manipulation and System control. They have been expressed in a level of detail that expresses the different interaction techniques associated to them. Removing the lower branches of those task models, those including modality and device dependant characteristics, the pruned tree showed a solution pattern for each 3D Universal interaction task.

Even that the use of pattern-based development life-cycle is contradictory to model-driven, as patterns are poorly structured or in many different ways. There has been some works [Gaff04] showing the potential of modelling patterns for task models and to reuse successful solutions. In appendix G, the 3D Universal interaction tasks patterns are expressed in terms of our task model. It is not the scope of this thesis to go beyond this description since most issues related to pattern-based design is extensively addressed in the literature. The description is limited to the problem body, including the empirical background, and the task model (this accordingly to Alexander's notation,). This could be easily extended into more detailed pattern Markup languages such as PLML (Pattern Language Markup Language), as it was done in [Mont08].

4.2.2 Guidelines for Task Modelling

Guidelines are a way to provide designers with guidance on to accomplish their goals in a systematic way. Designs can be crosschecked against the guidelines in different ways: manually (M), automatically (A) or both (B).

(A) *Ambiguity avoidance.* The ambiguity problem [Pate99] indicates that a task model should be clear on what it is modelled. The idea is to avoid situations where a task has different task operators before and after that might lead to an ambiguity. When it is the case, a sub tree should be used in order to avoid any ambiguity. In the example illustrated in Figure 4-3, Tasks T2 .. T6 are located at the same level of the tree, but with different temporal operators. The task T2 enables T3 which becomes available right after T2 is finished. However, task T4 is related to T3 with a concurrency operator, which means that T4 can be performed before T3. If T4 is performed first, then it enables and passes information to T5, which might be a problem as T3 was not performed. The solution, depicted in the right hand side of Figure 4-3, consists in creating a new level when a new operator is introduced. Notice that 'T2 enables 'T3' which enables and passes information to T5'. Then, the whole first level is related to enabling operators. In this new scenario, even if T4 starts and is performed first, then T3 must be finished in order to complete 'T3' and enable the next task.

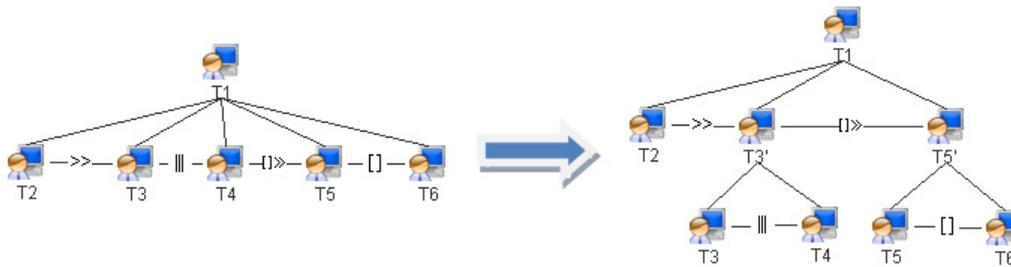


Figure 4-3. Ambiguity avoidance on task modelling.

(A) *Task Model Labelling*. Our empirical experience reveals that authors solving the problem of modelling tasks normally end in similar structures of the task model with minimal variations. However, the way authors label tasks may vary considerably. While authors can describe the tasks' names without following any rule – which is a sign of flexibility-, automatic transformation of task models to UIs becomes almost impossible considering the infinite variety of names that authors can choose for the task. From the taxonomy we propose a set of names (i.e., the task type, synonyms and sub-types) for naming tasks. Ideally, a task should be labelled with two elements: its name that reflect the task type and the object it manipulates (Figure 4-4). Task modellers could use for the task type any action from the taxonomy. Moreover, if the names are not considered correct or representative to the task, modellers will not be forced to use this methodological guidance to name the tasks, they may add a new name at any time. Our goal is to strive for homogeneity for the set of action types. The action types' names were kept independent from any interaction modality or IT. Modellers may however label their task using any synonym. For instance, users may label a task “Show results”, as the verb “show” refers to any sort of visual representation and the purpose of the taxonomy is to remain independent of any interaction modality. Consequently, for the task labelled “Show results”, its attribute action type could be “Convey” from the taxonomy. Notice that the relevance of keeping such naming for further concretization of the task model: if the final UI is vocal or physical, the action “Convey” is still representative while *show* is not since it may induce that a visual modality will be exploited.

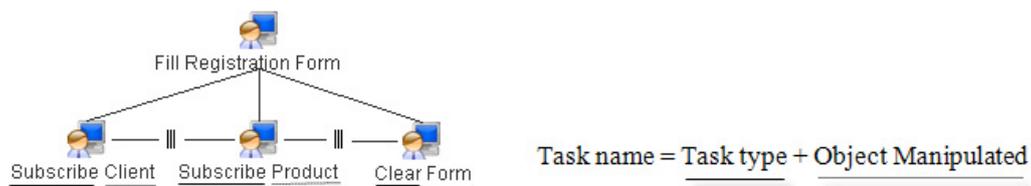


Figure 4-4. Guideline for task naming.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

(B) *Selection of action type.* So far, action types have been discussed in the taxonomy. Once the modeller names the task using the previous guideline, the next step consists in assigning a value to two task attributes: the *action type* and the *action item*. The action type can be automatically derived from the task type if the modeller chooses any abstract value for the action types, such as those found in the first column of Table 3-1. When it is not the case, a synonym can be used instead. The corresponding action type for that synonym is then assigned. Regarding the task item, the decision of which value must be assigned is based on the value that the task manipulates. In this case, we refer to the domain model of the problem and we have to look at the class(es) that the task manipulates, including the attributes and methods of interest. The assignation of this parameter is decided as follows:

- *Operation:* if the task manipulates a method, for instance “Insert a customer”.
- *Element:* if the task manipulates a data item that is bound to an attribute of a class belonging to the domain model, for instance “the name of a person”.
- *Container:* if the task manipulates an aggregation of elements, each element being an attribute of the same class or from several different classes. For instance, the attributes describing a book could be considered as a container, for instance “book title”, “book authors”, “book year”.
- *Collection of elements:* if the task manipulates an item that is a list of elements or containers. For instance, customer registration container (“customer name”, “customer address”) and the shopping list (“item reference”, “item description”, “item quantity”, “item price”).

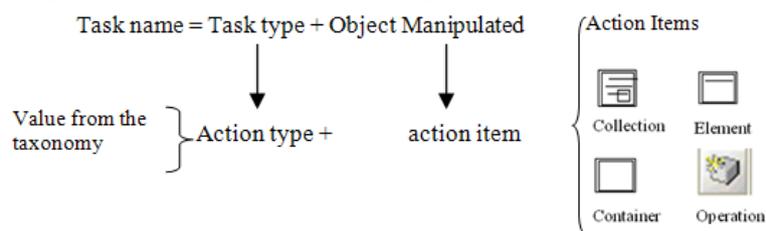


Figure 4-5. Guideline for task naming.

(B) *Selection of task category.* While authors concur on the need for separating the task types depending on the actor involved in the execution, it is not clear how to make this assignment. Lenorovitz et al. [Leno84] ended his review of the HCI discipline with a taxonomy of frequent interactive tasks. Johnsgard et al. [John95] separated the task categories into: user *interactive actions*, user *actions* and *system actions*. While *user actions* are more related to cognitive issues, *user interactive actions* correspond to the tangible manipulation of a system and *system actions* normally are

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

transparent to the user. The user does not know what is happening at the system level. Therefore, a selection has a different impact depending on who is performing the task. At the *user action* level means choosing something after considering several options, for instance, decide which flight ticket you want to buy. If we consider the same selection task at the *user interactive action* level, it means the interaction with the system to be decided, could be selecting a desired flight from a combination box. Finally considering the selection as a *system action* means that the system automatically will perform this action. For instance, the flight ticket could be bought using an agent that already know the end user's preferences and selects the best flight according to these preferences. Constantine [Cons03] introduces a taxonomy of *action types* and *action items* that enables a refined expression of the nature of leaf tasks (sometimes called *action tasks* or *leaf tasks* as they are usually located at the leaf nodes of a task decomposition tree). This expression qualifies a UI in terms of abstract actions to be supported. The taxonomy is twofold: a verb describes the type of activity at hand; an expression designates the type of object on which the action is operated.

(B) *Selection of task item.* The combination of task categories and UI action types (which itself consists of an action type and an action item) provides precise information for UI derivation. In order to understand the mean of the combinations between task item and user categories, a survey was conducted for each task type. As an example of the way task types have been investigated, Table 4-1 illustrates this for the task category "Mediate". For each task, there is a different meaning while being combined with different user's categories so as the task item they manipulate. The combination of these elements is useful for further concretization of the task, the whole set of action types can be found in Appendix D. The "Reinitialize" task refers to an item that either erases or cleans certain fields (e.g., a text field in a graphical modality, a text input in a vocal modality, or a gesture area in a tactile modality). This impacts the visual part but this action might have also some impact at the data level. At the data level, it implies restoring the default value. Reinitialize an element, a collection, or a container on a UI represents almost the same. However, the agent performing the task whatever that is (i.e., the user interacting with the system, the system or the cognitive decision making of the user), may hold different interpretations. System and user categories might have, in principle, no representation in the UI. When the task becomes interactive, the user might need an explicit mechanism in order to reinitialize the task item, i.e. an Abstract Interaction Object (AIO) [Vand93, Boda94] that will be concretized in further steps. In some cases the reinitialize task is implicit in the nature of some other task types: for instance, in a mailing website, creating a new user account normally involves the use of a form where users should fill a set of fields with per-

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

sonal data. Each element on the UI, unless something else is predefined, can be reinitialized by the user without the use of a reinitialize task for each element. It is always possible to erase any entry in a form and this does not mean that for each entry there will be a need for a supplementary task to specify that it can be reinitialized. This is what is usually assumed at the implementation level.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Mediate	 Collection	Compare products by price	Google search evaluating the best ranked pages to present the results of a query.	Analyze the data details (author, name, publisher, ...) of a book
	 Container	Compare side by side documents in word	Decide the layout of a slide when creating a new one	Compare a list of books
	 Element	Evaluate a video watched on YouTube	Evaluate the security risk of a password	Determine the date of a trip
	 Operation	Decide which operation to apply to a combination of CTRL keys	Propose different arrangement of the results of a query.	Decide which operation will be used with a special key on a joystick

Table 4-1. Mediate User Interface Actions.

The set of guidelines and task patterns that have been introduced here do not guarantee the usability of the future system but at least some consistency between proved solutions and new design for task models is established. Moreover, if the developers would like to perform some transformation on the task model, the concepts introduced in this section would contribute in further reifications of the 3DUI following where more guidelines, considering usability aspects, are applied in the selection of the corresponding concretization of the task and domain.

4.2.3 Running example: The Interactive Table task and concepts modeling

In this section a running example is show and hereafter will be used to illustrate the evolution of the method. The example corresponds to an interactive table. The user task, depicted in Figure 4-7, that the user can interact with a table, navigate through the room and interact with a screen. For the **navigate room**, the task model uses the identified task pattern (see appendix G) travel and Wayfinding. The **interact with big Screen** indicate as the **interact with screen** task refers to a turn on/off a screen than renders video or images.

4.3 Step 2: From task and domain model to AUI model

An AUI model can be generated automatically or produced manually from a task model following a set of heuristics. Various set of heuristics may fit this purpose depending on the type of AUI to be obtained: an AUI that reflects the task structure, an AUI minimizing navigation, an AUI compacting input/output. Some of these heuristics have been discussed in [Gonz06] for 3DUIs. Although this level is independent of any modality, some guidance is still desired on how AUI might be structured considering further reifications into concrete objects.

4.3.1 Guidelines for Abstract User Interface Modelling

(B) *Navigation incorporation.* Several metaphors have been introduced in order to display information or windows. If we imagine for instance a cube to render the different tasks as an Abstract Container (AC), then authors need to add inputs with navigation facets (Figure 4-8) in order to guarantee the cube transitions.

(M) *Facets selection.* Similarly to the task model, the AUI model uses the same attributes in to specify the UI action: action type and task item. In addition, the abstract level incorporates the facet concept. The “action type” attribute of a facet enables specifying the type of action that an AIC is allowed to perform. The “action item” attribute characterizes the item that is manipulated by the AIC. The AUI Model as well as the Task Model is independent of any modality of interaction. The set of possible AUI facets were introduced in the previous chapter in section 3.4. The *AuiInteractors* facets are: *DataInteractor* and *TriggerInteractor*. The *DataInteractor* is an aggregation of the different types of elements that interacts with the user to present (*Output*) or obtain data (*Input* and *selection*). The *DateItem* is a concept used for the abstract behaviour model with two aggregation relationships (*systemProvided* and *userProvided*) that represents the system and user provided data directly linked to the *DataInteractor*. This approach has some limitations with the simple values inputs and multiple or composed values. With this definition there is a clear separation between the data it manipulates (*DateItem*) and the type of interactor (*Output*, *Input*, and *Selection*). The abstract layer has few guidelines listed but the set is more than these two. More rules, described as transformation rules in Appendix A, are applied in order to transform a task model into a AUI model. The set of rules was conceived considering guidelines to ensure the creation of a usable guideline-based 3DUI in the next steps.

4.3.2 Running example: The Interactive Table Abstract User Interface Modelling

The task model of previous section (Figure 4-7) is reified into the AUI model, using the rules are similar as in the previous example so there is no need to explain the internal process of the tool. What is relevant is to notice, that even that we put in the task model the **make decision task** for wayfinding, at the CUI model this kind of task do not appear, as they are not part of the UI.

4.4 Step 3: From AUI to CUI

As depicted in Figure 4-6, AIOs can be selected based on the facet of the Abstract individual component (AIC), the action type and action item. Unfortunately, it is not enough while the action type and action item combined with the facet to properly select the AIO. An example can be used to clarify this situation. Assuming that the UI action type corresponds to a select of a collection of elements, then, several are the potential AIOs that can be used such as: combo box, radio button group, text fields. The problem became then on deciding the appropriate AIO depending on the context of use, the type of value to be selected, and the domain. For that purpose, the rest of UsiXML meta-models can be used. While models already exist and have been discussed in details in previous sections, our aim is on the use of them and to provide guidelines on the proper selection of AIOs. How to differentiate 2D and 3D tasks working on 2D or/and 3D objects? Another question is related to the final code. What is the appropriate representation of 3DUIs? Should the 2D desktop metaphor still be used or are there alternative visualizations or metaphors. Several attempts go towards defining a new toolkit of 3D objects [Andu06] which are natively appropriate to 3D applications. Again, this represents an advantage to have a predefined collection of such 3D-widgets, but then the interaction is reduced by what they offer natively.

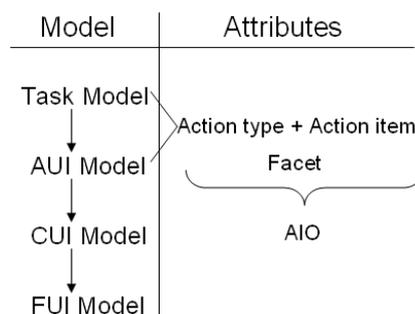


Figure 4-6. Overview of the AIO selection.

4.4.1 Guidelines for Concrete User Interface Modelling

The primary problem to solve at this level of abstraction consists in determining which mapping rules can be defined in order to transform an AUI into one or several CUIs. This is one of the most complex tasks in an MDE approach. This problem can be stated more specifically to 3DUIs as follows: ‘how to define spatial positions to place 3DUI elements or objects in a 3D scene’, which is not an easy task to automate due to the lack of semantic properties that define these spatial relations. Maybe the use of aforementioned taxonomy can be of some help to solve this problem as discussed in [Bowm04]. Designers can benefit from the taxonomy as the design space of UI actions is reduced to a set that is easier to handle. During a second phase of the process of developing 3DUIs, the task action can be mapped to a corresponding CUI. We have discussed the process of selecting 3D presentations based on the questions and answers method [MacL91] in [Kakl08]. We illustrate below how the taxonomy can be used for proper selection of the representation of the widget. The meaning of the links are: the darkest solid line (++) means strongly supported, dark solid line (+) means supported, solid line (~) means neutral, dash lines (-) means denied and dot lines (..) means strongly denied. The complete evaluation for AIO selection is in appendix C.

Unfortunately, the problem is not just a matter of widget representation, but also the selection of the widget itself. The possible mappings and guidelines to support the correct transfer from task model to UI widgets, as shown in [Boda94, John95], is also relevant. Because it is not enough just to use any arbitrarily selected widget, say a combo box for selecting a value, it is important to precisely define the conditions under which a particular widget may be selected: [Boda94] propose that the concretization of the select task must be based on the number of values to be selected (Table 4-3). While these characteristics are relevant in further transformations, following model-driven methods to derive UIs, examples can be found in [Limb04c, Pate99, Puer97], describing the task types using a good characterization of UI actions provides good basis for the concretization of the UI. As shown in Table 4-3, the select of a simple value can be mapped to a radio button group or a list box, the difference relies on the number of possible values to select. This characteristic could be part of the design, as used in Limbourg et al. 2004, the domain model in combination with the task model provide semantic information that can be further used on the specification of an AUI.

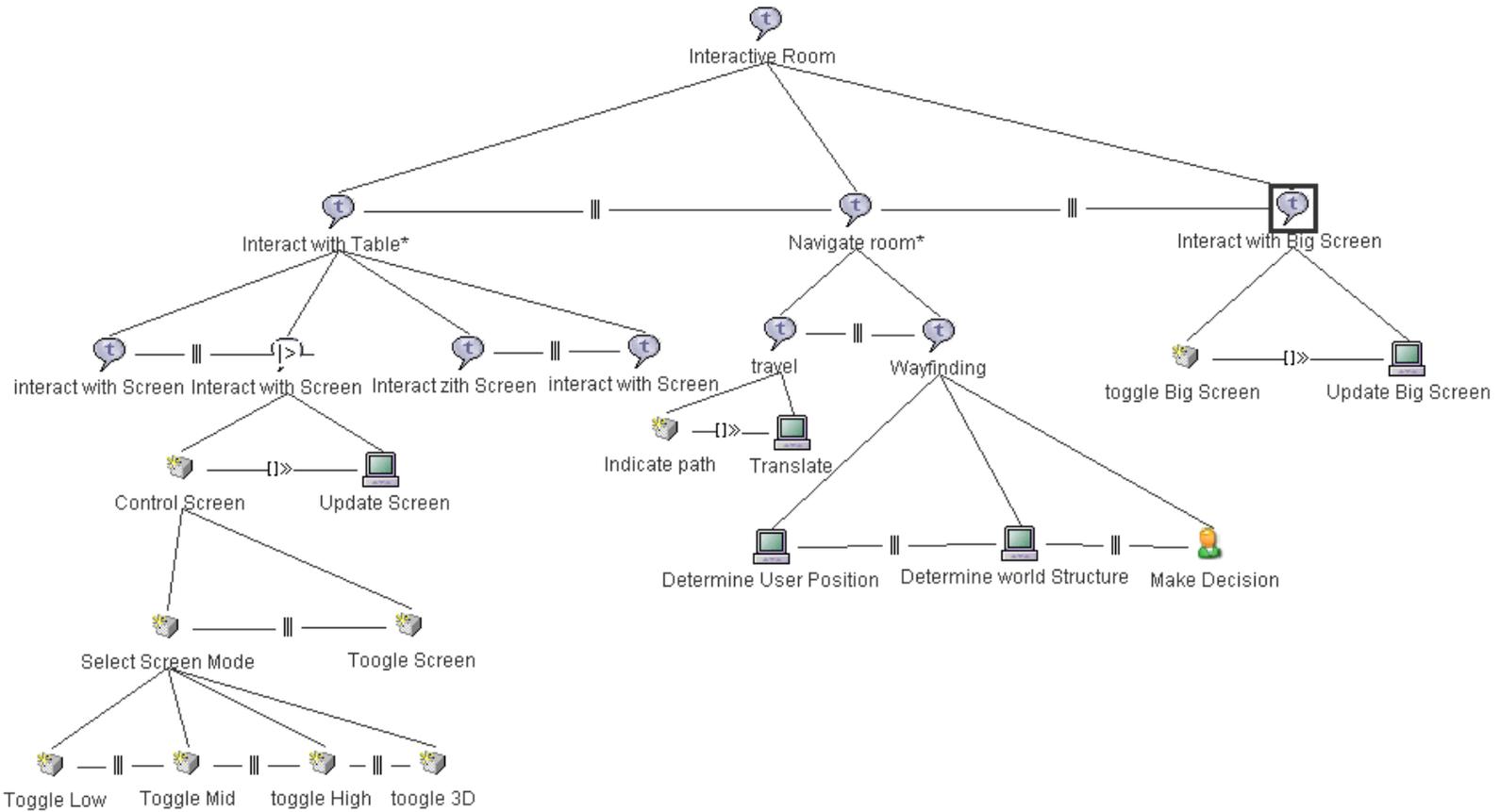


Figure 4-7. Virtual Office task Model.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

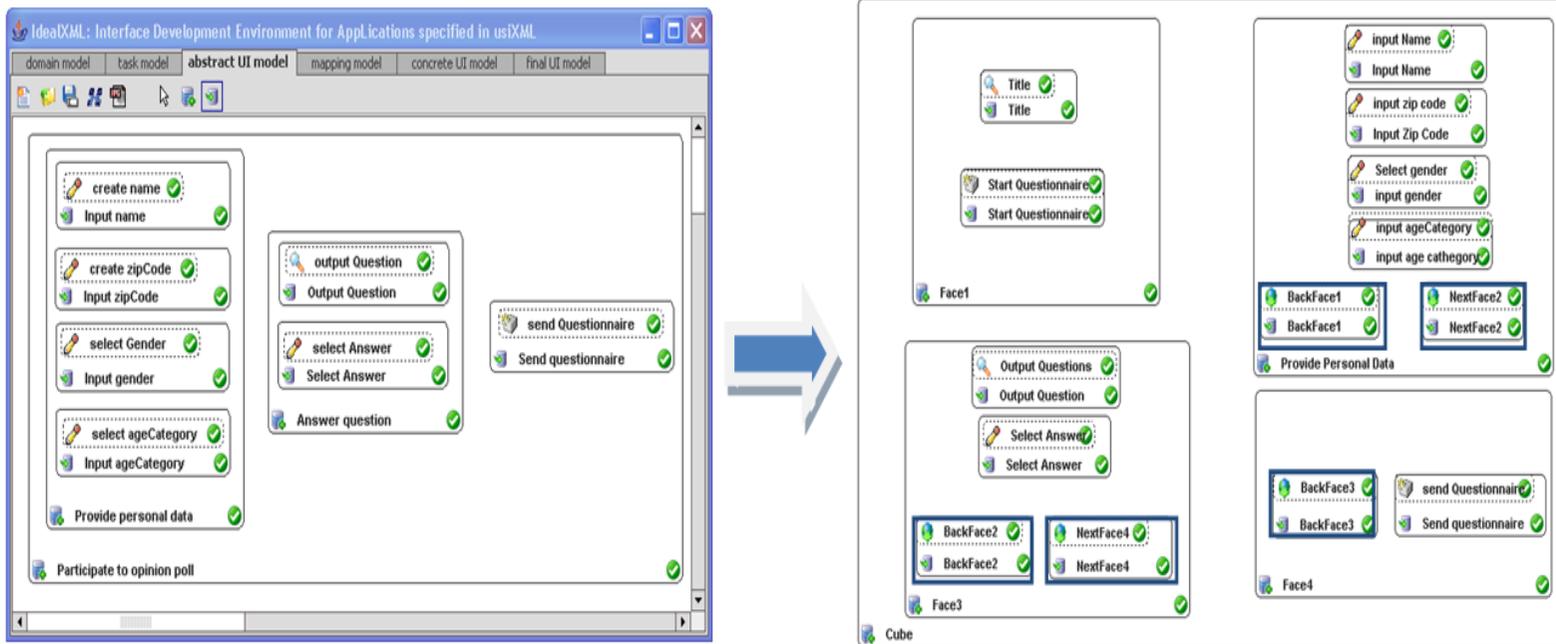


Figure 4-8. Adapting an AUI that further will need navigation facets.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

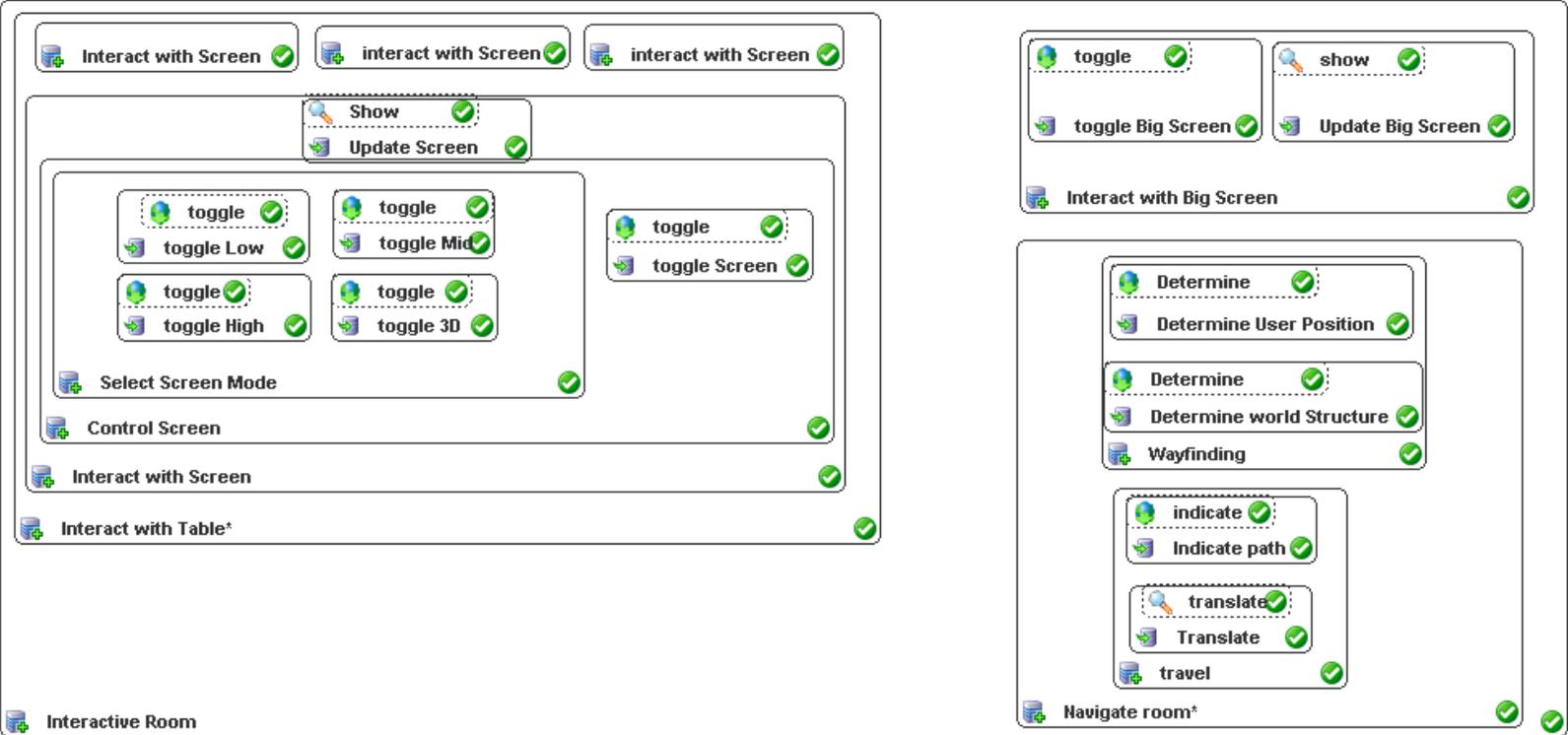


Figure 4-9. Virtual Office Concrete Model.

Question	Answer	Score	Options
What will be the representation of the 3D Toggle Button?	<i>Switch</i>	-	(1) 2D-3D Consistency
		--	(3) Easy to develop
		+	(4) Intuitional
		~	(5) Usability
	<i>Sphere</i>	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		~	(4) Intuitional
		~	(5) Usability
	<i>2D representation</i>	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	<i>Haptic</i>	+	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability

Table 4-2. Questions and answer criteria to select a toggle button.

	Representation
2D Representation	
Switch	
Sphere	
Haptic	

Figure 4-10. Graphical representation for a toggle button.

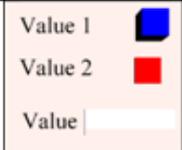
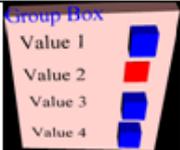
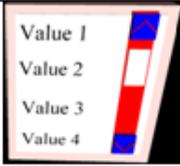
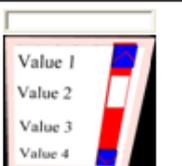
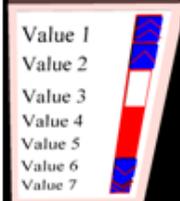
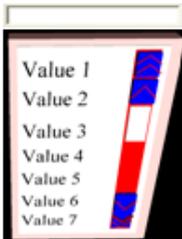
Number of values	Known Domain	Mixed Domain	Unknown domain
[2, 3]	 3D Check boxes	 3D Check boxes with text edit	 3D Text edit
[4, 7]	 3D Group box of check boxes	 3D Group box of check boxes with text edit	
[8, 50]	 3D List box	 3D Combination box	
[50, ∞]	 3D Scrolling list box	 3D Scrolling combination box	

Table 4-3. Which widget to select for which element of the domain model?

4.4.2 Running example: The Interactive Table Concrete User Interface

For our running example, the third step implies a transformational system that is composed of necessary rules for realizing the transition from AUI to CUIs. We won't consider in this example the attachment of objects to any surface, we just create a direct mapping, for each component and then in the high level editor each component is put in the corresponding shape. This sub-step involves the highest number of rules of all transformation sets as the different combinations of facet types, data types, cardinalities, are numerous. Table 4-4 provides the subset of rules applied in this case study. The designer can choose among the different alternatives provided by the rules.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

Abstract Interaction Component	Facet Specification	Information to take into account	Possible Concrete Interaction Component
“Navigation”	Navigation	The platform used, an internet Browser with any pug-in,	There is no need for any concretization of this task or any sub-task
Interact with big screen/ Screen	Toggle + Element (Screen/Button)	The big screen reacts on touch, the small screen react with a toggle button. This is also a design rule.	Toggle button, touch screen
“select screen mode”	Select attribute value + selection values known	The set of possible values are in subtask instead of a domain list of values.	A group of toggle button acting as a radio group.
“Toggle Low/Mid/High/3D”	Toggle + element		Toggle button
“Update Screen”	Communicate (Show) + Element (Image or video)	Attribute, data type, domain characteristics	An image component or video component

Table 4-4. Correspondence between AIO types and CIO types.

Physical constraints related to the size of the container and 3DCIC are considered with default values. Remember that if we could pass this specification to a high level editor allows us to arrange manually the objects. Ideally we expect to expand a virtual environment, such as Maya, in order to have import/ exports to our format, so; in this sense changes made in the toolkit can be track in the CUI model and backward, to have a consistent model at all the levels. The resulting specifications are obtained by realizing the above transformational development sub-steps. Figure 4-11, Figure 4-12, Figure 4-13, present a mock-up of the graphical UI corresponding to the CUI model.

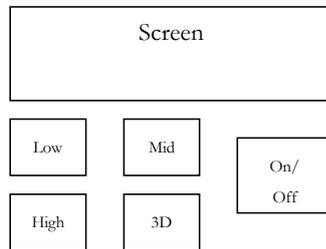


Figure 4-11. Mock-up of the Control Screen UI.

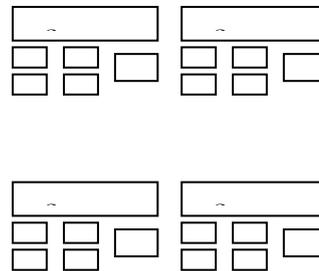


Figure 4-12. Mock-up of the Interacting table.

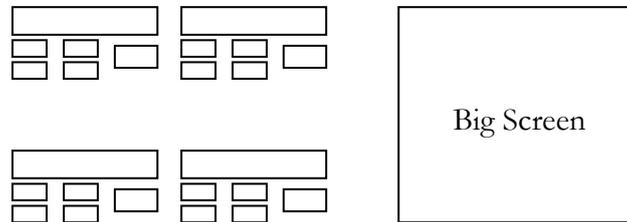


Figure 4-13. Mock-up of the Control Screen UI.

4.5 Step 4: from CUI to FUI

The resulting CUI 3DUI can be encoded in a software tool, such as: Vivaty studio, Blender, or any other, a list can be found in the state of the art section. The idea is not to start from scratch your models, this can be done but it is preferable to have a direct mapping in the tools and reuse existing work. Of course authors might produce their own objects if needed. The toolkits of which we rely to concretize 3DUIs that can be then rendered either in Java3D, VRML, X3D are discussed in the tool support chapter. The set of rules used are in appendix A. Applying the method is exemplified in the case study chapter.

4.5.1 Guidelines for Final User Interface Development

The final rendering of the 3DUIs At this stage of the development, any traditional usability method can be applied in principle. We are not aware of any system that performs usability evaluation directly on the code of a 3DUI (e.g., on VRML), although this could be a future avenue for automated evaluation. This is because at this stage, it is very complicated to analyze the code in a meaningful way. Some of the usability guidelines collected in [Bach04] could be evaluated at this stage, such as:

(M) *Realism of the objects.* Virtual objects should be similar as much as the real objects [Kaur98].

(M) *Compatibility with the navigation.* When the user is expected to navigate in a virtual world with a vast surface extension, it is important to let her navigate using different perspectives such as: egocentric and exocentric views [Gabb99].

(B) *Movement metaphors compatible.* The user might walk its avatar through the virtual world using the most appropriate metaphors, such walking, flying, virtual carpet [Gabb99]. Today most of the renders of 3D Web applications allows fast movements. However, the flying property or virtual carpet should be added to the avatar.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

(B) *Speed of the movement compatible.* Similarly the speed of the movement should be in harmony with the metaphor used, to fly faster speed than when walking [Gabb99]. The speed attribute of the avatar or virtual carpet can be checked to be different and with a $\text{virtualCarpetSpeed} > \text{avatar Speed}$.

(M) *The nature of the user movement compatible the human nature.* It is important that the user uses his body to interact in a virtual world in correspondence to the movements that they normally do [Kaur98]. This guideline is particularly important when gloves, head mounted displays or any other input device is used. However, it is applicable and relevant to Web application as the use of the keyboard and mouse should try to consider this issue as well. This is the case when using the augmented reality toolkit that can track the head movements so the viewpoint of the virtual world could be attached to the head movements.

(M) *Compatibility with the task and the guidance offered.* It is important that accordingly to the task some guidance should be provided [Kaur98]. This can be assured as the task model should be modelled considering the desired scenario. If it is a learning application then highlighting to guide the user must be explicitly determined in the task model then this information will be automatically considered when concretizing the 3DUI. Figure 4-14 depicts navigation in a virtual reality scene where the user is moving thanks to arm movements. The navigation direction (or pointer) is represented by a plot of dots that is moving according to the navigation. When the user requests some help, the pointer is transformed into a phone icon in the nearest environment in order to provide guidance.

(B) *Pointer should reflect when an object can be manipulated.* The pointer must provide a feedback when an object is different from the rest so that the user knows that there she can perform some action [Kaur98]. In many existing Web sites, a 3DUI is rendered that associates an object to a sensor so that this object changes when the sensor is activating this object (e.g., when the mouse pointer is over an object, this object may change). Of course, more sophisticated rendering could be associated to the pointer, such as voice feedback.

(B) *Objects and actions needed to perform a task must be available.* The user must be capable to perform the desired action using the objects needed to execute a task [Kaur98]. It is important to consider that a task model could incorporate aspects that might not be possible to perform on some applications, such as in a Web application. Then a task model incorporating those elements may trigger some warning of usability guideline violation, e.g. notifying that the task is not compatible with the available resources.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

(M) Actions available must be compatible with what the user expects. In a virtual world, the user expects to have some sort of actions available to perform her tasks [Kaur98]. This is the case of a user searching on the Web using a 3DUI [Kakl08] that combines capabilities for both sighted and sight-impaired people: when the user is sight-impaired, the actions available to the user are represented by haptic widgets (so-called ‘hapgets’) that the user can feel, differentiate, and manipulate. A sighted user may be interested by merely using the normal browser.

(M) Spatial organization of the virtual environment. It is important to keep the spatial distribution of the objects in a virtual world devoted to training or to be the mock-up of a place as similar as the real space. On the right part of Figure 4-15, a virtual office is used to mock-up a virtual world with an interactive table that is combined with some part of the reality (the corridor and the entrance depicted in the left part of Figure 4-15).



Figure 4-14. Guidance in virtual reality for requesting navigation help (Alterface.com)

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way



Figure 4-15 .Virtual office with an interactive table.

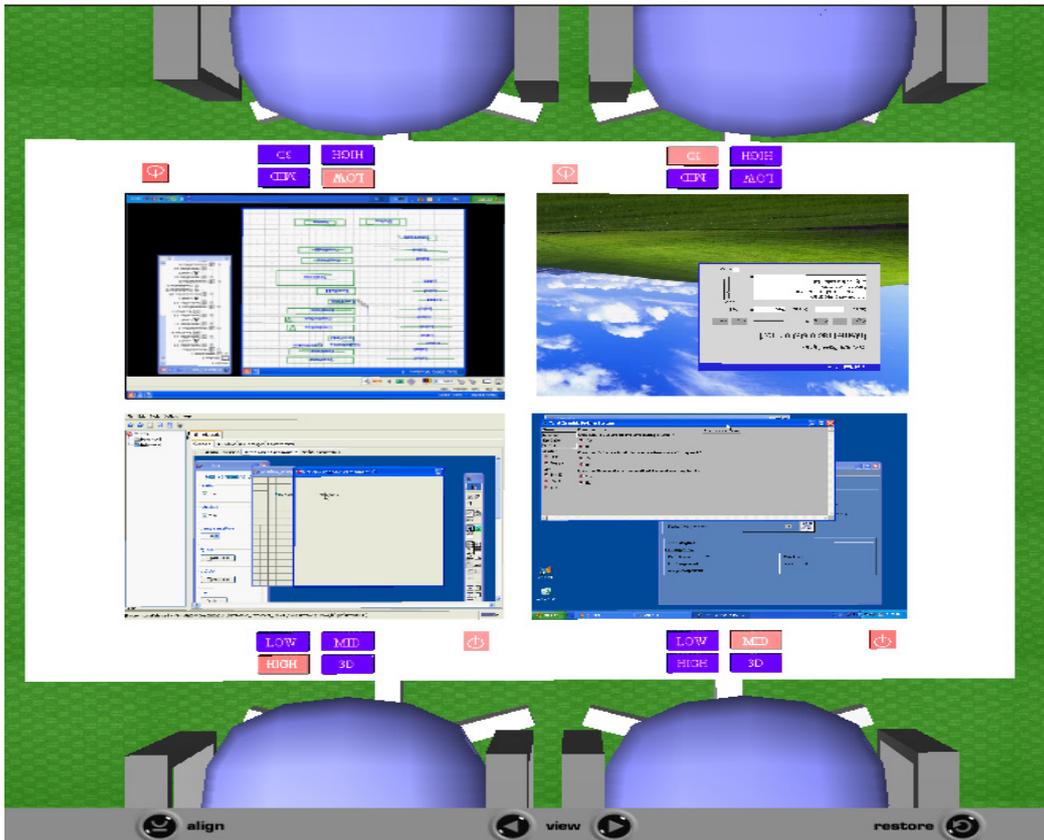


Figure 4-16. Top View of the virtual table rendered in VRML.

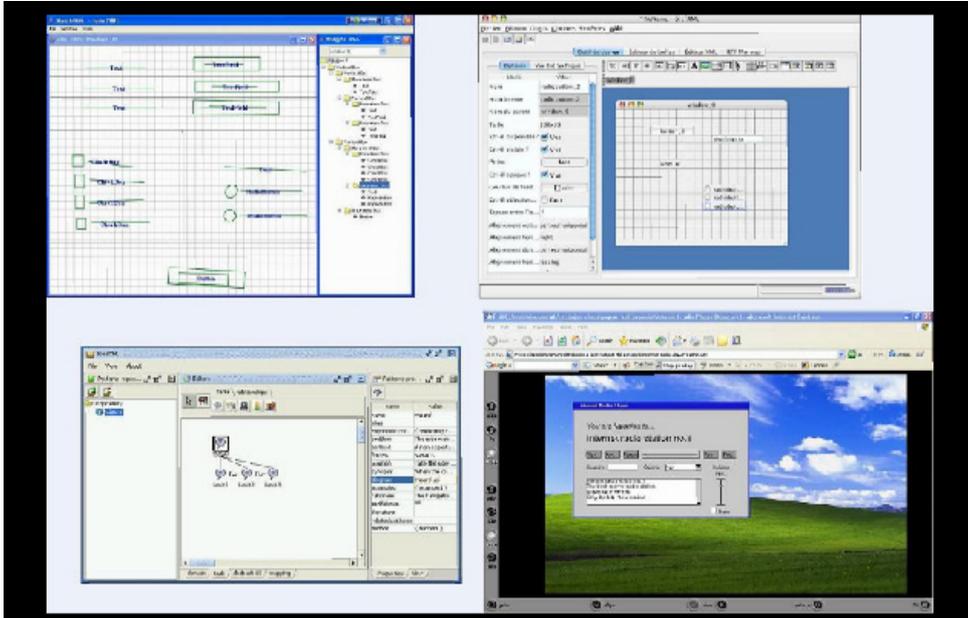


Figure 4-17. Big Screen rendered in VRML.

(M) *Spatial organization of the virtual environment.* Related to the previous guideline, this guideline refers to the need to represent a virtual world in a way that end users may easily discover some other areas related to the main one [Kaur98]. Authorities may want to know where the office is located and walk through the corridors before getting in the studio (left part of Figure 4-15).

(M) *Decoration appropriate to the context.* Decoration of the virtual world should be compatible with the context of use that is represented [Gabb99]. In Figure 4-15, the decoration is exactly the same as the building, carpet and walls colour, posters.

(A) *Provide avatar depending on the context.* There are certain tasks where an avatar may be useful to use [Kaur98]. For instance, in an electronic commerce Web site, an avatar may represent the customer so as to try items to buy. In an aircraft, such an avatar may become irrelevant.

(B) *Natural objects behaviour.* Objects manipulated in the virtual world must keep their natural behaviour unless something different is defined. By natural behaviour, it is meant that they should obey to physical laws, gravity laws, etc. This guideline can be linked to the behaviour model, where it can be checked that different behaviours are available. For instance, in the context of a space ship, objects may follow the gravity laws on earth but when outer space objects must float. By default, objects should follow the natural behaviour designed by the programmer. The realm of this behaviour cannot be tested automatically.

Thanks to our experience gained in using hapgets for various case studies, we also observe that several usability guidelines [Vand94] for haptically-enhanced widgets (or hapgets) could be identified and respected, such as, but not limited to:

(M) *The hapgets should be visually distinctive enough from each other.* For instance, the shapes used to represent the hapgets should be different enough to avoid any confusion. Two different hapgets, but having similar shapes, may be confusing visually, such as a circle and an ellipsis. This guideline is respected for the first set of presentations but not in the second where two hapgets are presented as a lozenge. Of course, a same hapget may appear several times (like in the calculator). Therefore there is a need to supplement the visual distinction by other means.

(B) *The hapgets should have colours that are distinctive enough from each other.* While having different shapes or volumes for different hapgets sounds appropriate, it is even better to reinforce this visual distinction by colour distinction [Vand94]. For example, spectrally-close colours should be avoided since they may generate no visual distinction. In addition, the different colours should be radically different when they are turned into greyscale so as to support distinctiveness for colour-blinded users.

(B) *The hapgets should be haptically distinctive enough from each other.* For instance, different hapgets do not have close values for their haptic section, for instance for attributes like BuzzEffect, ConstraintEffect, or inertialEffect.

(M) *Each hapget should be perceivable enough per se.* The values for the haptic section of each hapget should be beyond a threshold of perceivability. Otherwise, the haptic feedback is just ignored and do not provoke any emotional output [Salm08].

4.5.2 Running example: : The Interactive Table Final User Interface

For the running example the resulting FUI is shown in this section. The screenshot of Figure 4-17 shows the Big screen, Figure 4-16 shows the interactive table composed of four control screens (four). The navigation task is controlled by the Cortona player plug-in., which support the navigation with the mouse and keyboard, as input devices, the user just decide where to go.

4.6 Adding a new concept to the ontology

In the case that a new concept is going to be incorporated in the methodology particularly in the case of 3D widget, a series of steps are needed in order to support its integration.

Chapter 4. A Method for Developing 3DUIs in a Principle-based Way

First, the concept needs to be added to the ontology and for doing that there is a need to abstract its characteristics as: attributes, classes that compose the object, if more than one, and relationships with other widgets.

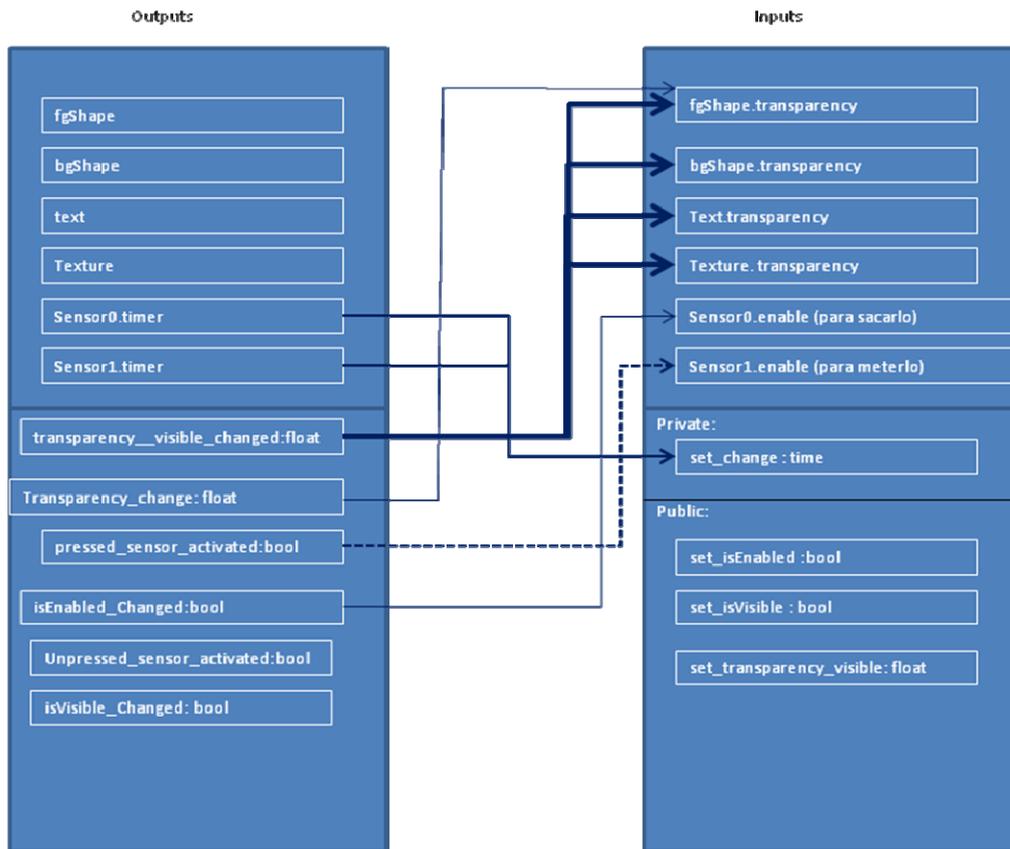


Figure 4-18 Implementation diagram of the toggle button

Once created in the ontology there must a correlation with its implementation. Attributes might have just one implementation attribute or several, and input and output events and their interconnection. In Figure 4-18 the toggle button is shown to illustrate this process. The different lines were used just to highlight the different routings. For instance, if the transparency of the button is changed then the shape and the text over the shape is affected. Then the lists of events that are public and private are defined, here we just show what we used for our internal understanding and developers are free to use their own techniques to model their implementations. For instance, in Figure 4-18 we graphically depict the routings that are affected enable a button method that changes the state of the event listener (sensors) and the transparency rate to graphically show which inputs and outputs are affected.

So far, the widget has been added to the ontology and its implementation developed. The next step is to identify to which tasks this widget is associated and consequently transformation rules are needed, a transformation rule(s) that allow reaching such new concretization of the task, as shown below.

New Rule (task Type, task item, abstract facet, domain of data manipulated ...) → New widget

4.7 Conclusion

This chapter is intended to present an alternative MDE-compliant method that is user-centred as opposed to contents-centric for developing 3DUIs. The approach considers a step-wise development life cycle where usability guidelines could be considered implicitly or explicitly at each development step. Also, the method proposes the use of automatic usability evaluation where guidelines can be stored as rules and subsequently tested during the transformation process. In general, model transformation holds the promise that each step could be achieved by applying a limited set of transformations, which are declaratively stated. It is a fundamental research problem to assess that the declarative power of such transformations, at least equal the procedural capabilities of algorithms traditionally used to produce a final UI. With this method, it is expected that the development of usable systems, like for the 3D Web, will be facilitated.

The potential advantages of using this method are: *modifiability* (if there is a change in a model then the 3DUI changes accordingly), *reduced complexity* (as it provides ways to address complexity, huge quantity of code, as well as the reliability), *safety critical* (models are needed to ensure the behaviour of safety critical systems), *formality* (the use of a formal specification technique is extremely valuable, because it provides non-ambiguous, complete, and concise ways of describing the behaviour of the systems), *rigorousness* (the development life cycle of the 3DUI involves the same level of rigorousness that is typically used in software engineering), *reasoning* (it is possible to reason about the models used to specify the 3DUI, such as automated evaluation, simulation, verification of properties).

Still this method presents some shortcomings. It is likely that the model transformations of large systems will be more complex to discover and to apply, so it is not clear if the solution is computationally feasible and scalable considering the amount of operations needed to perform graph transformations. Finally, empirical data coming from an evaluation of the methodology is required. Such data would emphasize the value added by this methodology to develop 3DUIs. An evaluation is planned as part of the future work.

Chapter 5 Towards a UIDL for 3DUI

So far, models and the approach to be MDA compliant have been presented. In order to be fully-MDA compliant, this works need a User Interface Description Language (UIDL). Some environments may includes models, and a transformational approach but do not have a genuine modelling language behind. It is not just because there is a XML language that a genuine modelling language may exist [Vand08]. A genuine UIDL must be strongly defined based on a trilogy (semantics, syntax, stylistics) [Sott07]. Offering a XML language does not necessarily assures to rely in this trilogy. The resulting UIDL (Figure 5-1) is in line with the requirements, based on the method and specify the ontology.

In this chapter a review of the literature of existing UIDLs is presented that were reported in [Guer09b]. Based on this review UsiXML was selected to be the UIDL for this thesis and the arguments that sustain this decision are exposed. Then, the UIDL trilogy (semantics, syntax and stylistics) is presented in details for the language.

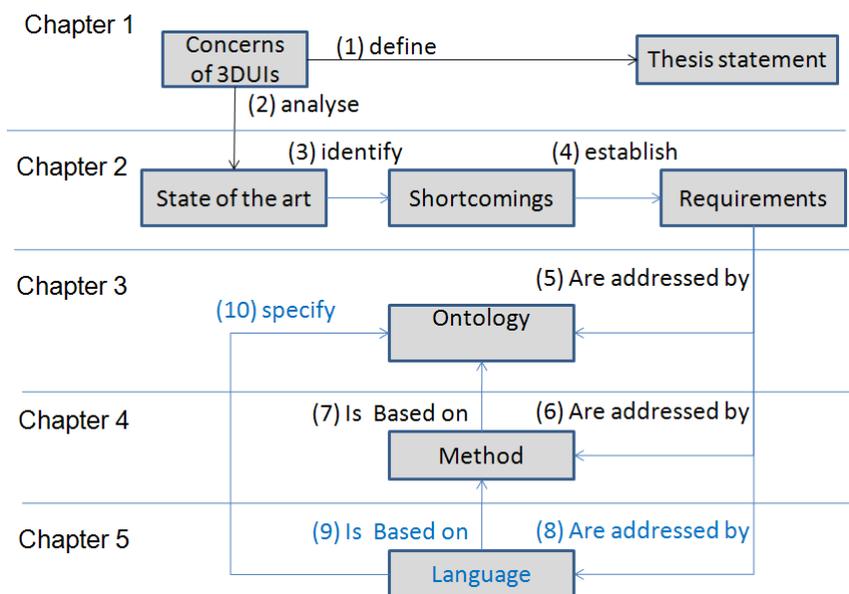


Figure 5-1. General Schema for UIDL derivation.

5.1 UIDL Selection

In the state of the art we introduced a comparative analysis on existing UIDL languages. UsiXML language was selected to support our model-driven approach due to the following motivations:

- UsiXML is structured according to different levels of abstraction defined by the Cameleon reference framework [Calv02]. The framework represents a reference for classifying UIs supporting a target platform, and a context of use and enables to structure the development life cycle into four levels of abstraction: task and concepts, AUI, CUI and final UI (FUI). The models described in previous section can be added in the corresponding slot.
- UsiXML relies on a transformational approach that progressively moves from the level to the FUI.
- The transformational methodology of UsiXML allows the modification of the development sub-steps, thus ensuring various alternatives for the existing sub-steps to be explored and/or expanded with new sub-steps. Thus helping in the exploration of different solutions for the final representation of the 3DUI and its evaluation in the cognitive model.
- UsiXML has a unique underlying abstract formalism represented under the form of a graph-based syntax.
- UsiXML allows reusing parts of previously specified 3DUI in order to develop new applications. This facility is provided by the underlying XML syntax of UsiXML which allows the exchange of any specification. Moreover, the ability to transform these specifications thanks to a set of transformation rules increases their reusability.
- The progressive development of UsiXML levels is based on a transformational approach represented under the form of a graph-based graphical syntax. This syntax proved to be efficient for specifying transformation rules [Limb04c] and an appropriate formalism for human understanding.
- UsiXML ensures the independence of modality (Ability to model a UI independent of any modality) thanks to the AUI level which enables the specification of UIs that remains independent of any interaction modality such as physical, graphical, vocal or 3D interaction.
- UsiXML supports the incorporation of new interaction modalities thanks to the modularity of the framework where each model is defined independently and to the structured character of the models ensured by the underlying graph formalism. It has the property of being extensible to new modalities.
- UsiXML is supported by a collection of tools that allow processing its format; it has the property of being machine processable of involved models.

- UsiXML allows cross-toolkit development of interactive application thanks to its common UI description format. Many UIDLs have been conceived that contain different features and focus on different levels of granularity. From the review a set of shortcomings have been identified:
- The set of widgets cannot be expanded, just the target language if a corresponding interpreter is developed.
- The most supported software is available the most restrictive is the language, for instance UIML.
- The language is context dependent. For instance AUIML is dedicated to accessibility issues and should probably be used only in these circumstances.
- The language focuses on a particular aspect of the MDA framework. For instance AUIML is today more part of the internal processes of IBM than in a complete suite of tools.
- The language is not intended to be a genuine and complete UIDL, consequently the language lack of expressiveness. For instance, XUL is mainly intended to support different viewing capabilities that are required to be supported by different computing platforms.
- The language is copyright protected. For instance, XIIML is protected by copyright by the XIIML Consortium. Any software that is XIIML compliant can consequently be distributed only if the future user of this software already possesses a XIIML license.

To the above identified advantages a more general one is added: whenever we would like to submit an extension of an existing language there is no guarantee that the Consortium in charge with that language will consider it as UsiXML consortium did.

5.2 UIDL Semantics

The 3D UIDL will be defined by relying on the formal specification language UsiXML, therefore ensuring rigorousness and formal analysis and handling of these specifications. The 3DUIDL specifications will be expressed in a new version of UsiXML (User interface eXtensible Markup Language) that will be adapted to 3DUI. In this way, it will be possible to conduct multiple formal analyses on the 3DUI specifications such as model checking, properties verification and validation. Semantics (in Latin letters *semantikós*, or significant meaning, derived from *sema*, translated as sign) is the study of meaning, in some sense of a term. For the language models are depicted as Meta-Models using UML class diagrams. This has been already showed in Chapter 3.

5.3 UIDL Syntax

This section specifies its syntax as a support of the semantics of the ontology introduced in Chapter 3. Syntax is often opposed to semantics: while the latter pertains to what something means, the former pertains to the formal structure in which something is expressed. On the one hand, the semantics of our ontology is defined by employing UML class diagrams. On the other hand, the syntax of the UsiXML language has an XML-based format structure, which allows describing sets of data with a tree-like structure [Stan08]. Figure 5-2 illustrates how the ontological concepts defined in the previous section are transformed in a UsiXML specification, which considers XML Schemas [W3C01] for the definition of valid XML elements. For this purpose manual transformations (T1) are applied in order to produce UsiXML XML Schemas from the UML class diagram description. Objects resulting from the instantiations of class diagram concepts are further transformed (T2) into UsiXML specification. Finally, the UsiXML specification is validated by the corresponding XML Schema.

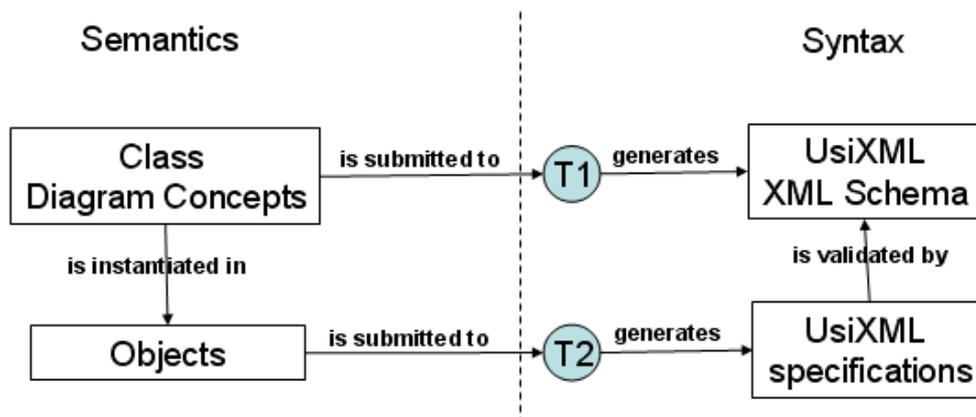


Figure 5-2. Generation of UsiXML specification.

From [Stan08], the following figures we illustrate how instances of a set of class diagram concepts are submitted to transformations T2 in order to obtain UsiXML specification. Following the path a class is instantiated in an object then the object generates a UsiXML specification.

A *class* becomes an *XML element* and *class attributes* become *XML attributes*: Figure 5-3 exemplifies how an instance of the *verticalProfile* class is mapped into an XML element with the associated attributes.

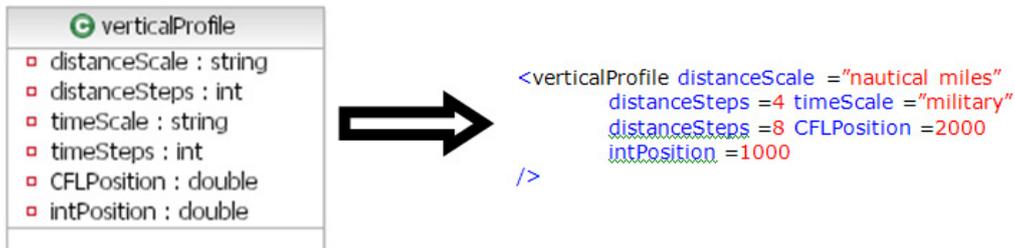


Figure 5-3. Correspondence of a class in UsiXML specification.

A *relationship class* and the associated *source/target classes* are transformed as follows: an *XML element* specifying the name of the relationship and *source* and *target XML elements* corresponding to the source and the target of relationship, respectively (please simplify the sentence). Figure 5-4 exemplifies how a *graphicalTransition* relationship between two elements (i.e., a source represented by the *plan button* and a target represented by a *plan window*, the *graphicalTransition* has an effect dissolve. This example illustrates the current version of the AHMI that uses dissolve effects to navigate from window to another. The UsiXML specification is as follows:

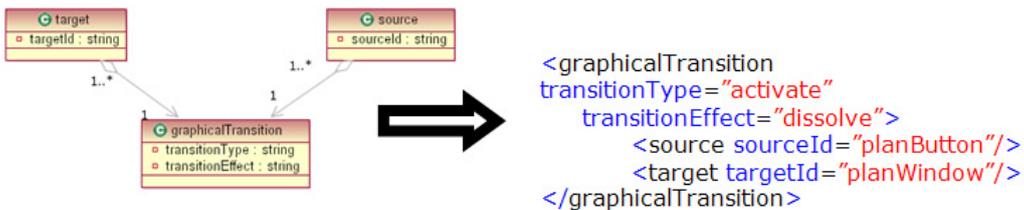


Figure 5-4. Correspondence of a relationship class in UsiXML specification.

Inheritance relationship class is transformed into an *XML element* for which the value of the *type* attribute takes the name of the subclass. In addition, the attributes of the parent class become XML attributes of the created element.

Figure 5-5 presents two objects of two different classes (i.e., *input* and *output*) that inherit attributes from the same superclass (i.e., *facet*). For each object an XML element is created. The attributes of the subclass instances (i.e., the *inputDataType* and *outputContent*) become XML attributes of the corresponding *facet* element.

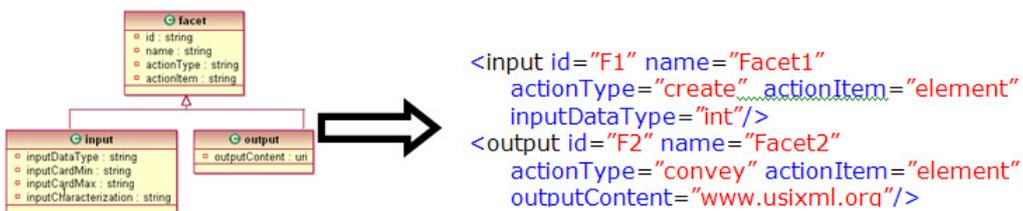


Figure 5-5. Correspondence of the inheritance relationship into UsiXML specification.

5.4 UIDL Stylistics

This visual syntax is mainly used to depict almost all the models defined in the ontology; there is a graphical representation for: the task model, the abstract models, graph declarations and transformation rules. The main diagrammatic characteristic, so as the software tools that support them, are depicted in Table 5-1.

5.5 Conclusion

In this chapter, a survey of UIDLs was conducted with the goal of selecting a UIDL adequate for extension towards 3DUIs. We hope this analysis will help in understanding and comparing the components of different UIDLs in a systematic way –their strengths, limitations, and appropriateness for use. There is currently such a large number of UIDLs available that choosing among them can be time consuming and difficult to do, this comparison can assist UI designers in choosing a language suited to their purposes.

There is a plethora of UIDLs that are widely used, with different goals and different strengths. On one hand we have software vendors UIDLs and, on the other hand, there are free license UIDLs to use; also some of them can support just one platform and others are multiplatform. Some of them (as WSXL or SunML) need a few tags while others (as UsiXML) have many of them. Also, some of them belong to the research area while others belong to the commercial. Even more, some of those UIDLs have standardization. Considering all those characteristics may seem hard to pick up one. We believe that this choice is more dictated by the goals to be pursued if one decides to adopt one of these UIDLs rather than only the different criteria that have been compared.

Different groups of people can define different vocabularies: one group might define a vocabulary whose classes have a 1-to-1 correspondence to UI widgets in a particular language, whereas another group might define a vocabulary whose classes match abstractions used by a UI designer. UsiXML is structured according to different levels of abstraction relying on a transformational approach to move among them. It ensures the independence of modality thanks to the AUI level. It is supported by a collection of tools that allow processing its format.

We chose UsiXML because it is: open, accessible, expansible, model-base compliant. The language was detailed as a trilogy: syntax, semantics and stylistics, thus assuring to be compliant to language engineering.

Chapter 6 Software for Supporting the Methodology

So far, the methodology for 3DUI development has been presented. This methodology relies on three axes: models (ontology), approach and language. The next step is to present the set of software for supporting the methodology. The set of software is in line with the requirements as depicted in the general schema evolution of this dissertation (Figure 6-1).

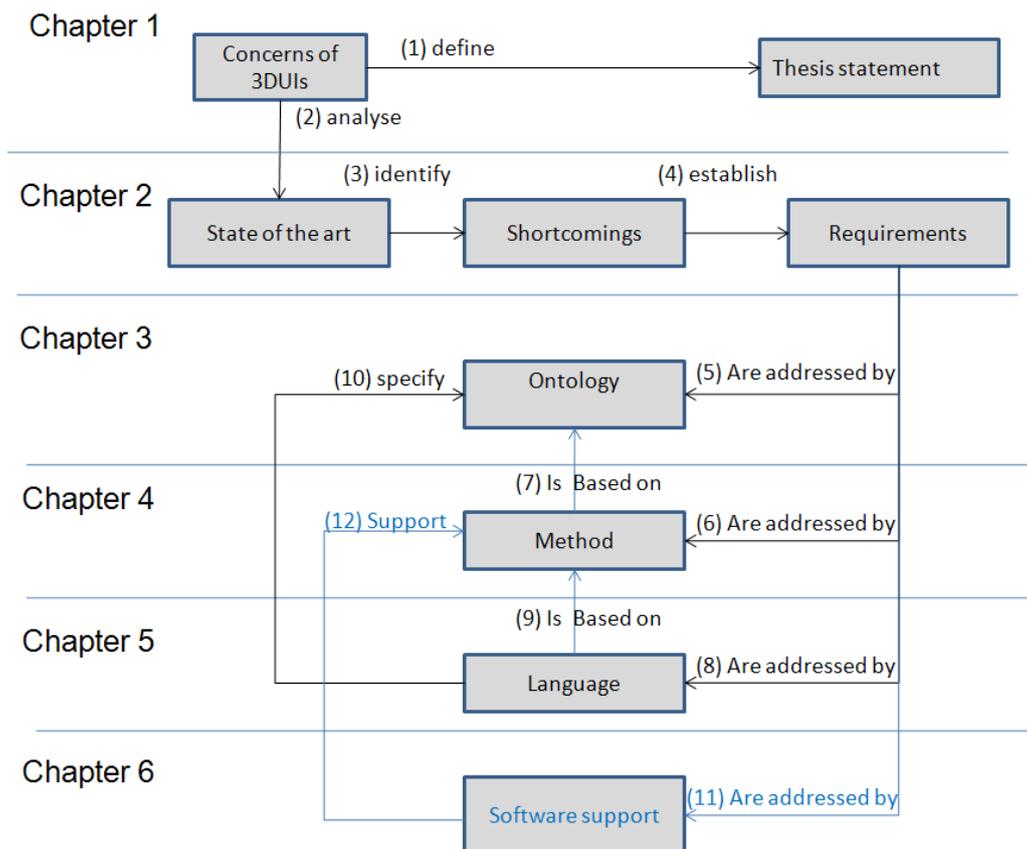


Figure 6-1. General Schema for UIDL derivation.

As discussed in [Limb04c, Szek96, Schl96, Puert97] the set of software tools required to support the development of 3DUIs includes:

Chapter 6. Software for Supporting the Methodology

- *Model editors* assist a designer in constructing the models. These tools consist in syntax editors, form based tools, or visual builders. Some model editors maintain a textual specification consistent with a graphical representation.
- *Design critics* provide a designer with quality assessment facilities. Models capturing explicit properties of the artefact are an ideal representation to perform evaluation.
- *Implementation tools* translate a specification into a representation that can be used by a compiler, an interpreter or an interface builder.
- *Transformation tools* provide support to the designer to edit, store and execute model transformation rules.

Finding the right tool is a trade-off between six main criteria [Shne04]:

1. *Part of the application built using the tool.* Some tools only support building the presentation part of the application; others also help with low-level interaction, and some support general programming mechanism usable in other parts of the application as well.
2. *Learning time.* The time to learn the tool varies.
3. *Building time.* The time required to build a UI using the tool varies.
4. *Methodology imposed or advised.* Some tools strongly impose a methodology for building the application, such as building the visual part first and connecting it into the reminder of the application afterwards, whereas other tools are more flexible.
5. *Communication with other subsystems.* Applications frequently use databases, files located on the web, or other resources that, when supported by the building tool, simplify the development.
6. *Extensibility and modularity.* Applications evolve, and the new applications may want to reuse parts of existing applications. Supporting the evolution and the reuse of software remains a challenge. Level-4 tools and application frameworks, including MDA, inherently promote good software organization, but the others usually lead to poor extensibility and modularity.

An overview of the set of software tools used to support the methodology is shown in Figure 6-2. Task and domain models, along with their mappings are edited in IdealXML. The AUI model can be generated automatically from these specifications and edited using IdealXML as well. Model to model transformations are supported with TransformXML or Yate. The CUI model can be edited using Alice, Vivaty Studio, or directly coding on any text editor. In the reminder of this chapter, the set of software tools used to support our methodology is presented. In most cases such tools are the result of previous works. However, when applicable a discussion on the contribution coming from this work is emphasized.

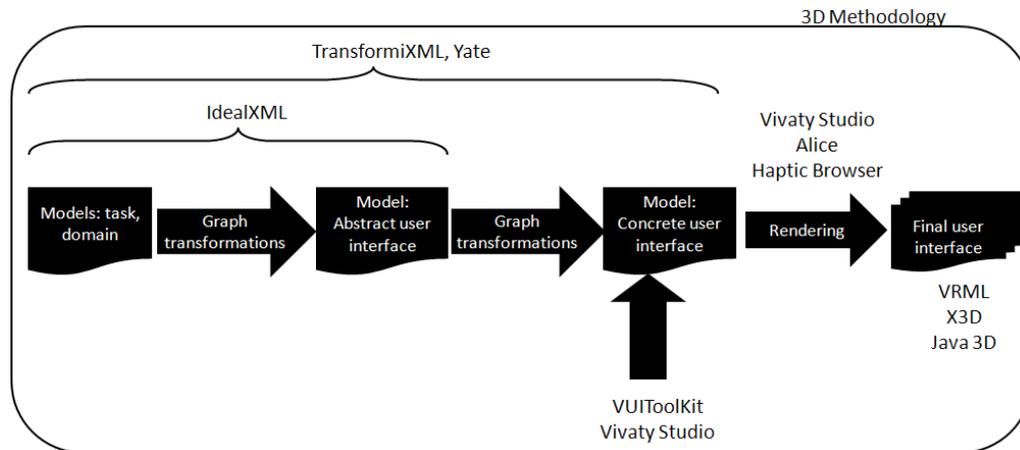


Figure 6-2. Software chain for supporting the Methodology.

6.1 Model Editors

From step one to three there are two tools used to design task and concepts (domain model) and generate its UsiXML corresponding code, using IdealXML [Mont05]. Similarly, IdealXML is used to define the abstract user. The stylistics of this tool has been presented in Section 5.4.

In IdealXML the domain model is represented as a UML class diagram (Figure 6-3a). The task model is a hierarchical structure of task and relationships (Figure 6-3b). The AUI defines abstract containers and individual components, two forms of Abstract Interaction Objects by grouping subtasks according to various criteria, a navigation scheme between the container and selects abstract individual component for each concept so that they are independent of any modality. An AUI can also be considered as a canonical expression of the rendering of the domain concepts and tasks in a way that is independent from any modality of interaction. An AUI is considered as an abstraction of a CUI with respect to interaction modality. At this level, the UI mainly consists of input/output definitions, along with actions that need to be performed on this information. This step is also supported by IdealXML (Figure 6-3c). Finally, the mapping model containing a series of related mappings between models or elements of models (Figure 6-3d).

The case studies chapter will present more details on how this tool was used in practice.

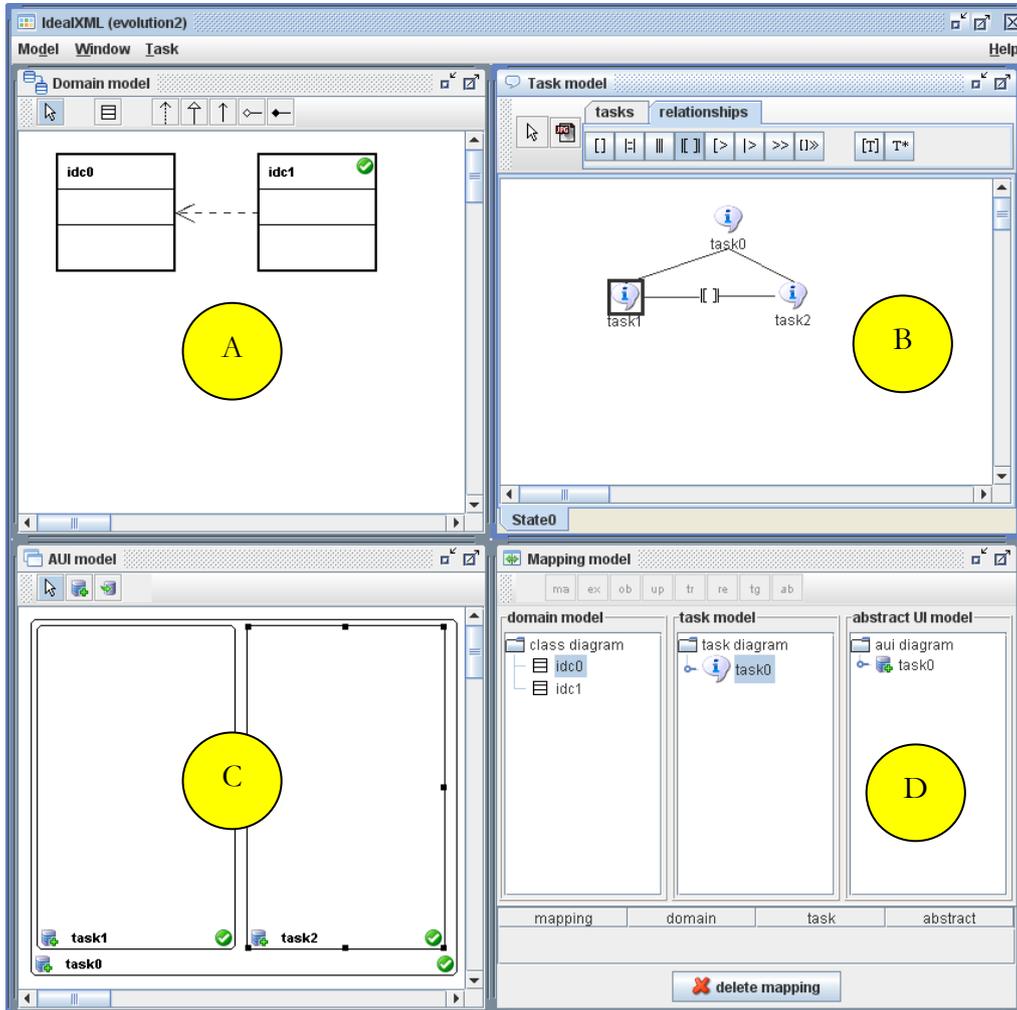


Figure 6-3. Task, Domain, Abstract and Mapping Models specification in IdealXML.

6.2 Design Critics: Guidelines evaluation

In Chapter 4, usability guidelines were introduced for each development step. In this section, we show how such usability guidelines can be evaluated in different ways: manually (M) when the guideline can be evaluated only by human evaluators, automatically (A) when the guidelines can be tested by software, or both (B).

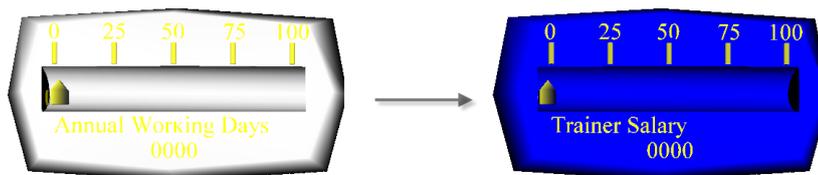


Figure 6-4. Adaptation based on ergonomic rule.

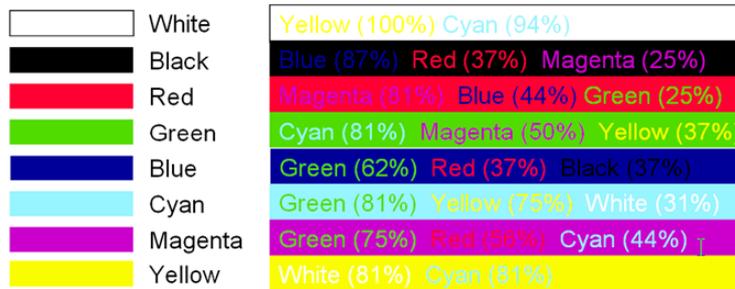


Figure 6-5. Bad colour combination.

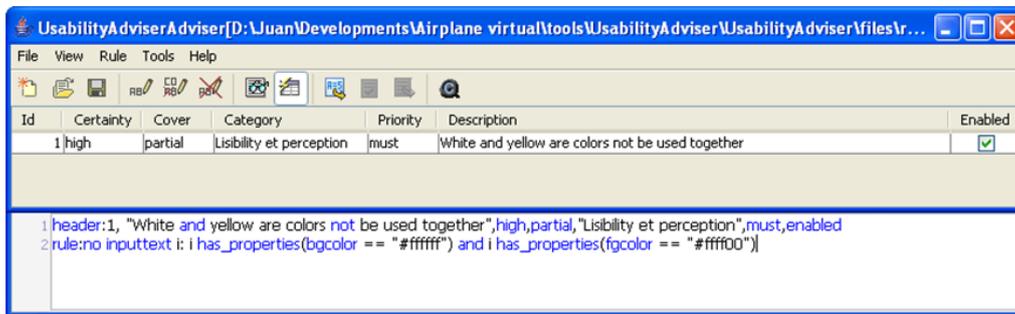


Figure 6-6. Usability Advisor Interface.

The complexity of implementing software that automatically evaluates such guidelines is beyond the scope of this chapter. Therefore, we refer to the literature for this purpose, for instance [Vand99, Vand05]. Nevertheless, automatically addressing usability has been already explored for 2DUIs with the Usability Adviser [Vand06]. This software determines the usability of any UI specified in UsiXML, a user interface description language by parsing its specifications and examining them against a set of usability guidelines encoded in GDL (Guideline Definition Language). This software expresses usability guidelines as logical grammars. For example, a usability guideline that selects appropriate colour combinations (Figure 6-5) can be written for the usability advisor as follows:

$$i \in \text{Slider} : \neg (\text{SliderColor}(i, \text{white}) \wedge \text{LabelColor}(i, \text{yellow})),$$

then if this occurs a change could be to chose automatically the most recommended colour, that seems to be the blue as background of yellow letters, Figure 6-4. Usability guidelines can be added, removed, and edited from a configuration file that stores the logical expression. This software is primarily used at the CUI level where AIOs [Vand94] are mapped to CIO by relying on information hold at more abstract levels. The main window of the Usability Advisor is reproduced in Figure 6-6. Usability guidelines for 3DUIs applications have been introduced in several research papers, for instance: for guidance during navigation in augmented virtual environments [Gram06, Smit04]; to design objects for reduced spaces [Kaur97] and user interaction in virtual environment [Kaur99].

Chapter 6. Software for Supporting the Methodology

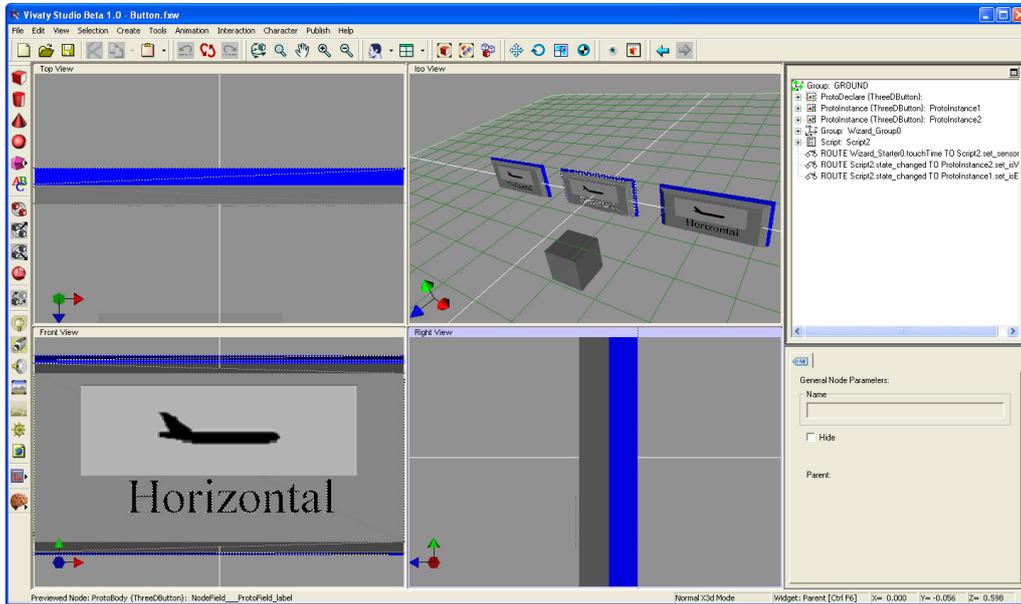


Figure 6-7. TDCIOs toolkit in Vivaty Studio.

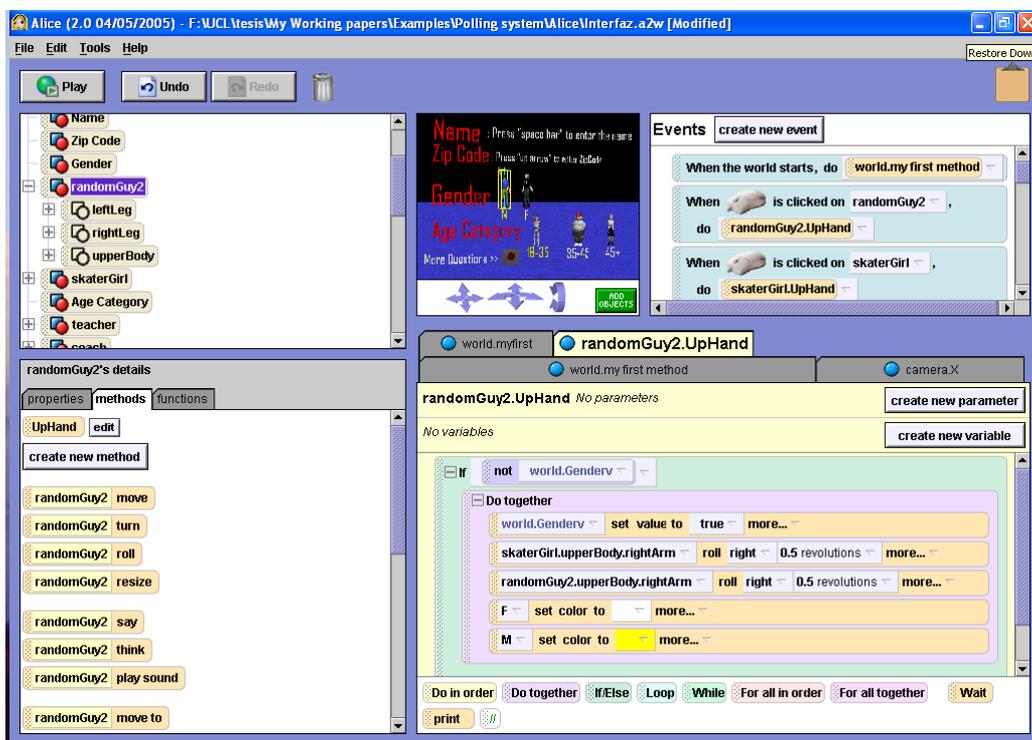


Figure 6-8. Alice Environment.

Even if many similar usability guidelines exist for the Web, many of them can also be used for 3DUIs after some adaptation. In [Boda94], the problem of selecting a widget from information contained at the abstract level is addressed. For instance, a selection task can be mapped onto different widgets depending on: the number of possible values, the number of values to be selected, the domain whether is known or not or a mixture of both. From the compilation of guidelines from [Bach04], a set of guidelines for automated evaluation was extracted. Notice that for Web 3D applications there are no immersion and sophisticated input devices available in general. Here are some significant examples:

Objects availability. Objects associated to a task must be available to perform the task [Kaur98]. Objects in the virtual world are directly associated with the task model so that in theory they will not be missing.

Object's highlight for guidance. Relevant objects for the task must be highlighted to guide users on their task [Kaur98]. The domain of the problem provides some sort of information of the purpose of the virtual world. If an object was chosen to be highlighted for a specific purpose, this can be checked by referring to the highlighted object's property at the concrete level (CUI).

Object transparency feedback. Objects in the virtual world which are solid should provide a feedback to the user in order to let the user know their limits [Kaur98]. This property can be checked automatically with the rule: if a solid property is defined, then the feedback property must be defined.

Different exploration facilities available. The application must provide facilities to explore the virtual world in different modalities [Gabb99]. Most existing browsers provide means to jump into predefined viewpoints that might be relevant for the user or useful if they are lost.

Many more guidelines are applicable, not just for 3DUI for the Web in general. Again, this is beyond the scope of this chapter.

6.3 Implementation tools

In step three, our need is on a software tool that could support the design and manipulation of 3DUIs, for this purpose we proposed the use of Maya software, which is a very powerful commercial editor for 3D content. A concrete UI can be exported from its definition in UsiXML and imported to Maya (due to its compatibility with X3D specification), or any other toolkit, for instance: Anark, Max 3D, or a non-commercial tool, such as Blender or Vivaty. In the review of the state of the art we identified that the two most advanced and extended toolkits are Max 3D and Maya, as they have too many plug-ins for importing / exporting in several formats, some of them of our interest, such as VRML and X3D.

Chapter 6. Software for Supporting the Methodology

Actually we draw our 3DUIs in Maya and Vivaty Studio then we export them to VRML and X3D (the plug-in produces low results for complex interfaces). Then using Vivaty Studio editor we added the behaviour to the interfaces. Our first experience with Maya showed that this approach was tedious. Our example has more than 30 thousand code lines (most used for the geometry), and it was difficult to identify shapes and add behaviour directly into the code X3D. The use of a X3D-compliant editor help I the improvement of this process and even more in the creation of a toolkit of TDCIOs that can be reused emulating a visual studio editor for 2DGUI development.

A second high level toolkit that we used to model the UIs is Alice that generates Java 3D content. In Alice it is possible to reuse existing objects but not to create new ones. Then, CUI concepts need to be added manually and is limited to those existing in the tool. When a different behaviour is needed then the method editor can be used. In Figure 6-8, right bottom, the code corresponding to select the gender, whether Masculine M or Feminine F, is shown. The method presented is for the click on the Male option, which means select the male in the 3DUI. Using a state variable, called world.Genderv, to identify if the male is selected or unselected, when unselected a set of actions are triggered (Do together). These actions correspond to the male falling his hand and the female rising its hand. e selected value to the state variable world.Genderv, the following second and third lines corresponds to rise the hand for the man and down the hand for the woman, finishing with the modification of the text objects F and M, changing their colour to white and yellow respectively.

Alice is an easy to use toolkit very useful for academic purposes, its main goal is to teach how to program object oriented applications. Also, covering VR spectrum apparently it looks that it catch the attention of major industries, as the creators of Sim City, the famous real life simulation game, will donate their characters, which look so realistic to enhance the presentation of Alice. Apparently there will be more news about Alice in the near future. In order to allow a direct mapping from UsiXML specification we need to explore how the Java 3D API is structured and how Alice format is structured. The advantage of Alice is that they do not just have primitive objects but also a Gallery of predefined objects. Unfortunately, Alice have a reduce set of objects dedicated to the UI control, button, switch, text and no more. Finally, the last toolkit that we used to generate the FUI is the VUIToolkit [Moli05]. This library of widgets does not have neither an editor to design the 3DUI, nor an automatic generation from the concrete model to the FUI in VUI components.

6.4 Transformation tool

Model engineering (i.e., a discipline that is concerned with the development of models) is part of the numerous solutions proposed to overcome with the increasing complexity that developers must handle to produce software. Model-driven development (MDD) is an OMG (www.omg.org) initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. It advocates that software development should be guided as much as possible by the construction, and refinement of software models at various levels of abstraction. Four principles underlie the OMG's view of MDA for UIs:

1. **Models** are expressed in a well-formed unified notation and form the cornerstone to understanding software systems for enterprise scale information systems. The semantics of the models are based on meta-models.
2. A formal underpinning for describing models in a **set of meta-models** facilitates meaningful integration and transformation among models, and is the basis for automation through soft-ware.
3. The building of software systems can be organized around a set of models by applying a series of **transformations** between models, organized into an architectural framework of layers and transformations: model-to-model transformations support any change between models while model-to-code transformation are typically associated with code production, automated or not.
4. **Acceptance and adoption** of this model-driven approach requires industry standards to provide openness to consumers, and foster competition among vendors

Not all model-based UI development environments or development methods can pretend to be compliant with these principles [Vand05]. If we apply OMG's principles to the UI development life cycle, it means that models should be obtained during steps of development until providing source code, deployment and configuration files. MDD has been applied to many kinds of business problems and integrated with a wide array of other common computing technologies. Considering MDD of UIs [Limb04a] complexity related to the number of transformation needed to support this process has been found in the literature as a major issue [Limb04d]; In Figure 6-9 how graphs transformations area articulated when following MDD method. Each development path (for instance, forward engineering) is composed of development steps, the latter being decomposed into development sub-steps (for instance, from abstract to concrete models). A development

Chapter 6. Software for Supporting the Methodology

sub-step is realized by one (and only one) transformation system and a transformation system is realized by a set of graph transformation rules.

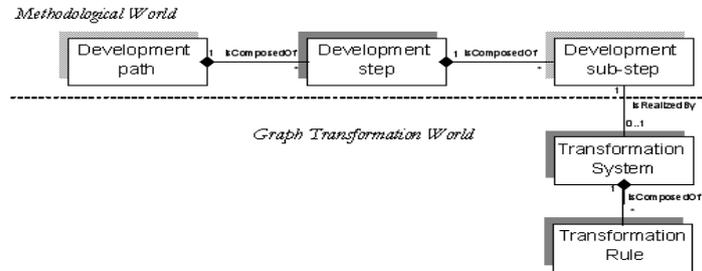


Figure 6-9. Link between graph transformations and transformational development of UIs.

The amount of transformation rules of a rather simple system grows up to two hundred graph transformation rules. Then, usable systems are needed to encode transformations rules. Also, provide facilities to extend existing sets of graph transformations. Recently, we have been working on this issue. On the one hand, trying to use existing tools to encode graph transformation rules applied to MDD of UIs. On the other hand, building from scratch customized tools [Limb04e, Stan08] addressing the same problem.

In [Gonz08a] a survey of transformation engines was presented. This section summarizes the results of that survey that lead to the selection of the tool to support transformations between the different steps of the method. From this review it was founded that no best tool exists. The selection can be based on author's preferences. In our case, graph transformation was selected as a method to support model to model transformations; more details are in appendix B. The selection of graphs transformations were based on the fact that graph grammars:

- Are rather declarative and provide an appealing graphical syntax which does not exclude the use of a textual one
- Are based on a formally defined execution semantics based notably on pushout theory, for which many proofs have been provided (completeness; confluence)
- Allow to describe transformations with the same vocabulary as specification models in a very consistent manner and for all development steps
- Provide extensions (i.e., conditional graph rewriting, typed graph rewriting) to check important properties of the artefact that is produced after a transformation
- Offer modularity by allowing the fragmentation of complex transformation heuristics into small, independent chunks. The fact that graph rewritings have no-side effects facilitates this modularization.

6.5 Haptic Browser

The complexity and the skills required to develop Graphical/Haptic (Multimodal) User Interface (GHUI) stress for a toolkit where native GHUI are provided to deploy a GHUI application.

In this section the procedure that was followed for the implementation is described. This work was done in collaboration with Nick Kaklanis and has been reported in [Kakl08a, Kakl08b]. The main concept is that users give a URL as input to the application, then some necessary transformations are executed and a UsiXML file that describes the specific web page is generated. Using this UsiXML file and a template database which contains all the 3D components implemented in OpenGL, the application creates a 3D scene corresponding to the web page.

First of all, the HTML file is transformed to an XHTML file so as it can be parsed as an XML file. For this transformation an open source tool which called “Tidy” (<http://tidy.sourceforge.net/>) is used. After the XHTML file parsing, the application collects only the information needed about the components described in it and a UsiXML file corresponding to the XHTML file is created. At this point, the application managed to create a UsiXML file that describes the web page given as input. The final step is to parse the UsiXML file and show the components described in it in the 3D scene. This procedure is shown in Figure 6-10. For the creation of the template DB, first off all, “Blender” was used to design a 3D shape for each component. Using “Blender”, the correspondent Wavefront Obj files were exported and imported in an Perl application which is called “OBJ2OPENGL”.

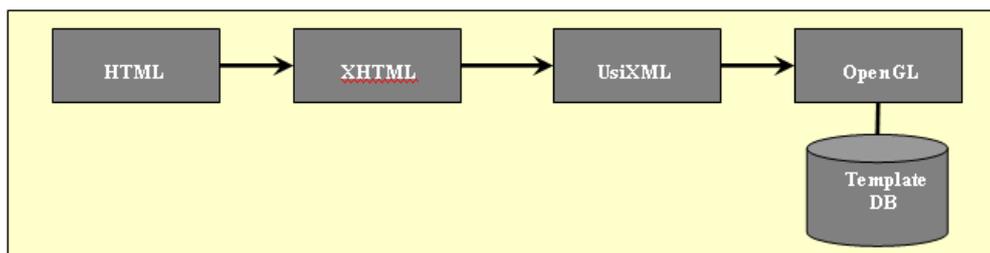


Figure 6-10. From HTML to a 3D scene.

The output of this Perl application was one C++ header file for each component that could be imported in a C++ project using OpenGL. Finally, this procedure resulted to a 3D scene created using OpenGL which included the 3D representation of all the 3D UsiXML components. All the above steps are described in Figure 6-11.

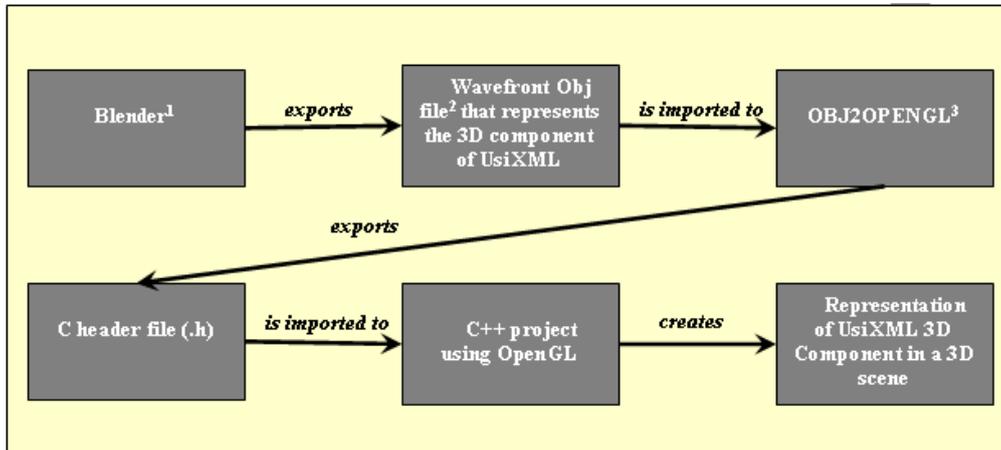


Figure 6-11. Steps of the implementation of the template DB.

The target group of users includes people with normal vision as well as visual-impaired people. Due to this, the representations of the components had to be meaningful for both categories. For instance, an image has no meaning for a blind person but the description of the image (alternate text) has. For this purpose, a text-to-speech engine was integrated to the haptic rendering engine so as to give the opportunity to blind people to hear what they cannot read. A speech-recognition engine which offers the opportunity of inserting text without typing was also integrated. Additionally, earcons were used so as each widget can be identified by the unique short sound which is heard when the cursor touches one of the widget's surfaces.

6.6 Conclusion

In this chapter a set of software tools that are used to support our methodology were introduced. This software is in line with the requirement of the methodology to be *machine processable*.

Edition of models is possible via IdealXML for task, domain, AUI, mapping models. For this tool there was no contribution in its conceptual design. However, the future direction of this tool must consider the new domain and AUI model. Also, the transformations considering the new set of canonical action types. Three software tools were used for the CUI model, Alice, Vivaty Studio and VUIToolkit. For none of these software tools a contribution on the implementation came from this work. It is mainly on its use and the way of creating components that can be reused in the context of this dissertation that we centered our contribution.

Chapter 6. Software for Supporting the Methodology

Also, a render engine to support haptic interaction was presented. This browser uses the meta-model introduced in section 3.5.7 for haptic support. The hapgets used in the browser are the result of the informal evaluation that is reported in appendix C.

With respect to the transformation system tool support, a comparison of transformation engines to support UI development was discussed. Four tools were compared. From our evaluation we noticed that it is difficult to find the perfect balance with the criteria. In particular, one of the goals of model-driven development of UIs is to support UI run-time adaptation based on transformation rules. Therefore, the performance is of concern. For the interest of this thesis, we can conclude that a systematic method is recommended to drive the development life cycle to guarantee some form of quality of the resulting software system. Not all of these technologies will directly concern the transformation involved in MDA. MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modelling language must be used. Any modelling language used in MDA must be described in terms of the MOF language to enable the metadata to be understood in a standard manner, which is a precondition for any activity to perform automated transformation.

Ideally, one software tool would have enclosed the support for the whole methodology. A solution closed to IdealXML where different software modules, corresponding to the edition and generation could the different models, could be opened and linked one to another. This is one of the future directions of the present work.

Chapter 7 Validation

The validation chapter (Figure 7-1) has for purpose to determine if the methodology satisfies the requirements introduced in Chapter 2. Also, the goal is to serve as a proof of the concept of the different principles introduced in the method, and to prove the feasibility of method through a set of case studies.

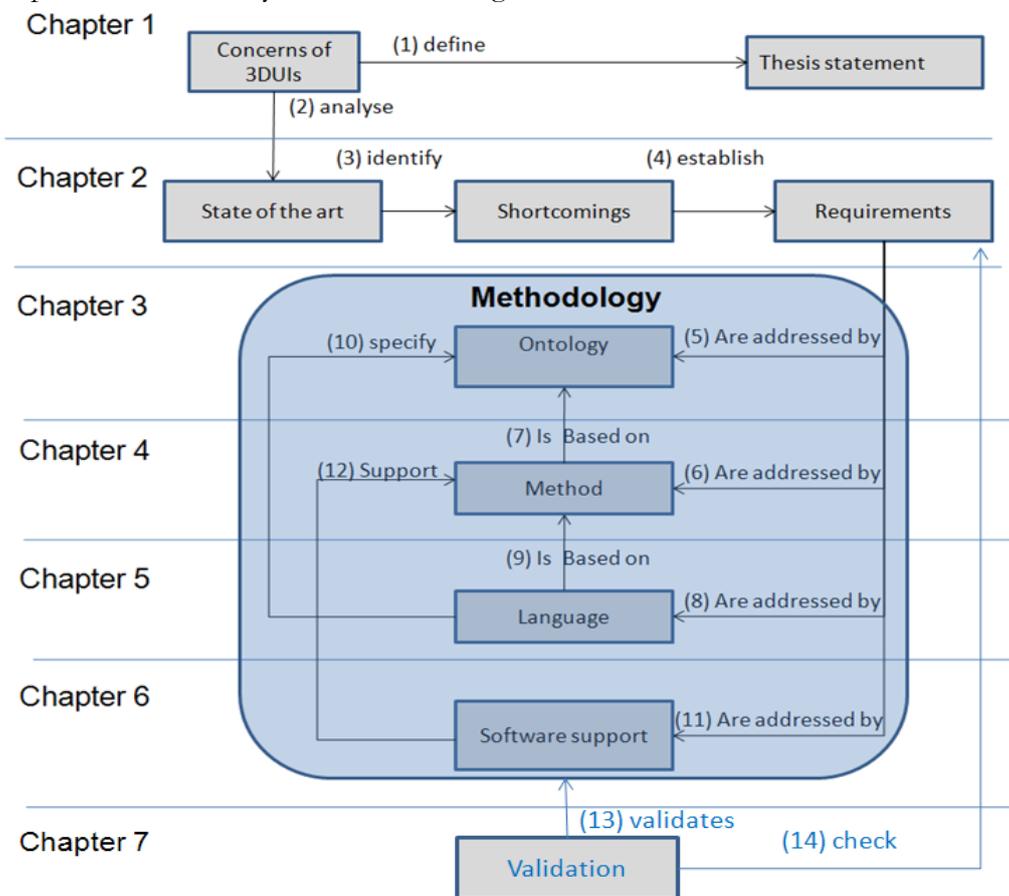


Figure 7-1. General Scheme of the validation.

7.1 Case Studies

This section applies multi-path development of user interface to four different case studies. The first three case studies are progressive in terms of complexity. The last one illustrates the use of the haptic browser. The case studies presentation relies on a series of illustrations showing how artefacts are progressively transformed according to various development sub-steps, steps, and paths. The process adopted to develop the case studies of this chapter consists of: (1) Build-

ing initial models. Such models have been edited with their associated editing tool. IdealXML [Mont04] has been used to edit the task and domain model. Most of the rules have been elicited prior to realizing these case studies by a theoretical analysis of development sub-steps as illustrated in Chapter 4. (2) Exporting resulting models to UsiXML and illustration.

The first case study is devoted to the development of an *opinion polling system*, a reasonably scaled example of a typical information system. The purpose of this case study is to show the feasibility of solving the problem of producing a 3DUI by applying a model-based approach. A forward engineering starts from the definition of the task and domain models to produce both an AUI and a CUI. The CUI is reshuffled by hand in Maya editor without changing the AUI.

The second case study is dedicated to the development of an *administrative application for a management school*. Even that the problem complexity is moderate it has been chosen to illustrate the design diversity that offer 3DUI. A set of parameters are used to calculate the number of trainers needed in a training academy.

The third case study is devoted to the development of a *flight navigation system for an aircraft*. The complexity of this system is high but is more related to the algorithms that compute aircraft behaviour during the flight, while all this already exist. We conduct an in depth study to provide a MDA to the development of the Navigation Display UI and namely using a 3DUI instead the 2DUI.

Finally, the fourth case study is dedicated to the *rendering of web site in the haptic web browser*. This section is aimed at showing how a 2D GUI can be rendered in a 3DUI enhanced with haptic feedback.

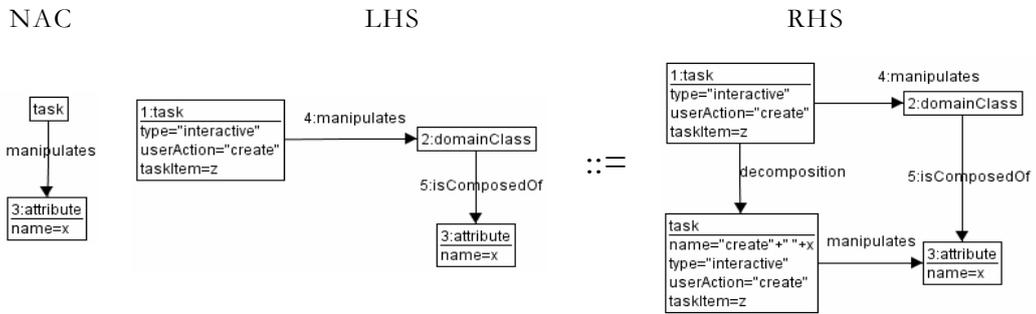
7.1.1 Case Study 1: The virtual polling system

This case study is devoted to the development of an opinion polling system, a reasonable scaled example of a typical information system. The development scenario is the following: a forward engineering path is applied from a definition of the task and domain viewpoint to produce both an AUI and a CUI.

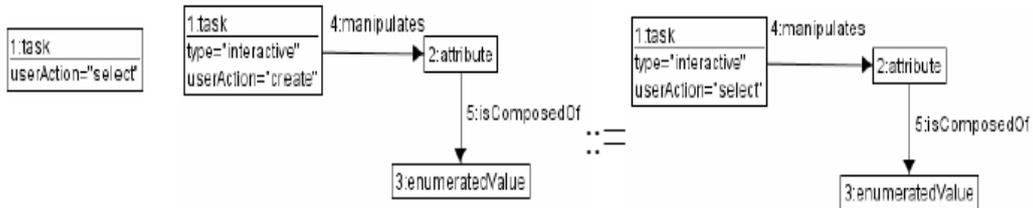
7.1.1.a Step 1: The task and domain models

The task model, the domain model, and the mappings between, are all graphically described using IdealXML tool [Mont05], an Interface Development Environment for Applications specified in UsiXML. Figure 7-2 (A) depicts the domain model of our UI as produced by a software engineer. A participant participates to a questionnaire. A questionnaire is made of several questions. A question is attached to a series of answers. The domain model has the appearance of a class diagram.

Chapter 7. Validation



Rule 7-1. Consolidation of the task model



Rule 7-2. Specializing a user action.

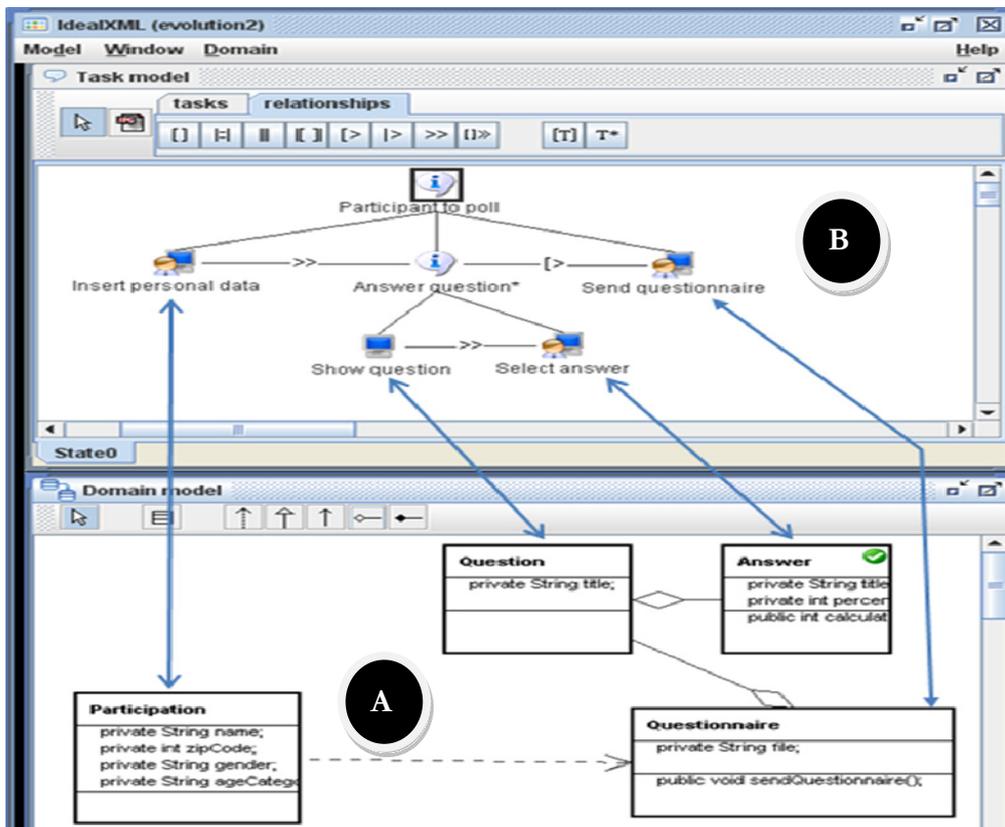


Figure 7-2 Mapping between domain concepts and task model

Figure 7-2.b illustrates a CTT representation of the task model envisioned for the future system. The root task consists of participating to an opinion poll. In order to do this, the user has to provide the system with personal data. After that, the user iteratively answers some questions. Answering a question is composed of a system task showing the title of the question and of an interactive task consisting in selecting one answer among several proposed ones. Once the questions are answered, the questionnaire is sent back to its initiator. All temporal relationships are enabling which means that the source task has to terminate before the target task can be initiated. The arrows between the two models in Figure 7-2 depict the model mappings, such as manipulates relationships between the task and the domain model as arrows. Provide Personal Data is mapped onto Participant class. Show Question is mapped onto the attribute title of class Question. The task Select Answer is mapped onto the attribute title of the class Answer. Finally, the task Send Questionnaire is mapped onto the method sendQuestionnaire of the class Questionnaire. The initial task may be considered as not precise enough to perform transformations. Indeed, the task Provide Personal Data is an interactive task consisting in creating instances of Participant. In reality, this task will consist in providing a value for each attribute of Participant. This could mean that the task model is not detailed up to the required level of decomposition.

Rule 7-1 is applied to the task and domain models. The Left-Hand Side (LHS) contains an interactive task (1) where the user action required to perform the task is of type create. This task manipulates a class from the domain model (2), which is composed, of an attribute that takes the value of a variable x . The Negative Application Condition (NAC) specifies that a task manipulates an attribute (3) whose name is stored in the same variable x . The Right Hand Side (RHS) specifies the decomposition of the task described in LHS (1) into an interactive task (2), which requires a user action of type create. Note the way they are named using a post-condition on their name attribute. The mappings between nodes and between edges belonging to the three components of a rule (NAC, LHS, RHS) are specified by attached numbers. The application of this rule on the task and domain model represented in the form of a graph G is the following: when the LHS matches into G and the NAC does not match into G , the LHS is replaced by the RHS, resulting a transformed graph G' . Therefore, Rule 7-1 decomposes the task Provide Personal Data into four new sub-tasks, each of them manipulating an attribute of class Participant.

Consequently, to the execution of this rule, four new tasks are created: create name, create zipCode, create ageCategory and create gender. Figure 5-1 shows the mapping model containing the mappings between the refined task model and the

domain model of the opinion polling system. Each of the four new sub-tasks will be mapped on the corresponding attribute of the class **Participant**, the rest of the mappings remaining the same. Due to the fact that “create” is a very general action type and that both **ageCategory** and **gender** attributes hold an enumerated domain, “create” can be specialized into “select”. Rule 7-2 is applied in order to achieve this goal. Rule 7-3 provides a default temporal relationship (set to enabling) when two sister tasks have no temporal relationship. The resulting task model is depicted in Figure 7-3.

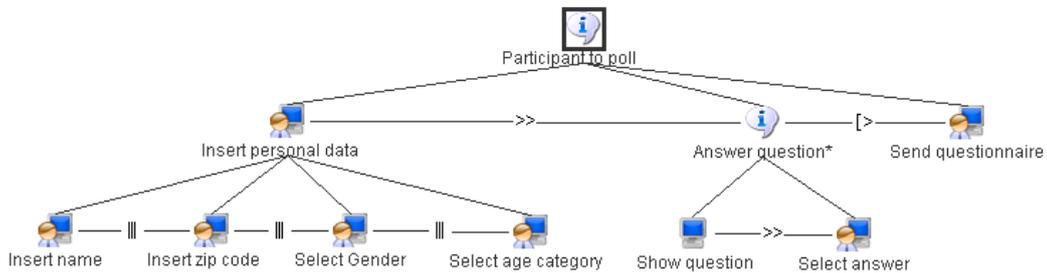


Figure 7-3. Consolidated task model of the polling system.

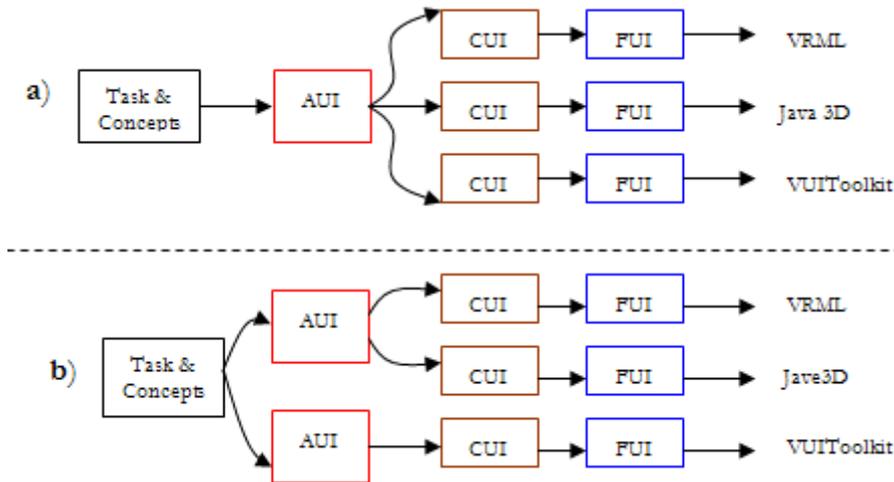


Figure 7-4. Possible Development Paths.

7.1.1.b Step 2: From The task and domain models to Abstract Model

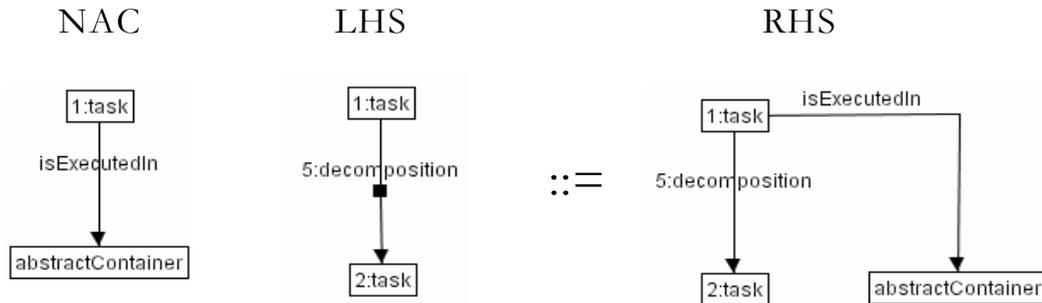
Different development paths can be followed based on the task model (Figure 7-4). In Figure 7-4 we show the different development paths that can be followed. For this case study path *b* has been selected.

7.1.1.b.1 Identification of abstract UI structure for AUI “A”

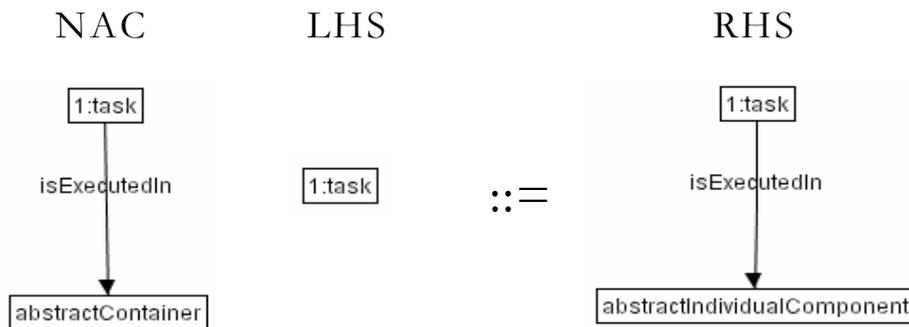
The identification of the AUI structure is ensured by applying Rule 7-3, Rule 7-4, Rule 7-5, Rule 7-6, and Rule 7-7. These rules essentially recreate the task model

Chapter 7. Validation

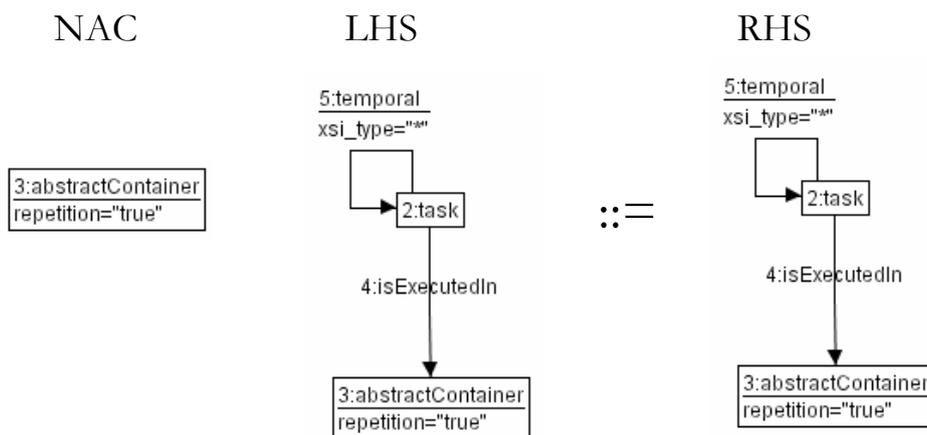
structure by a hierarchical decomposition of abstract containers and abstract individual components (interactors).



Rule 7-3 Create an AC for task that has task children

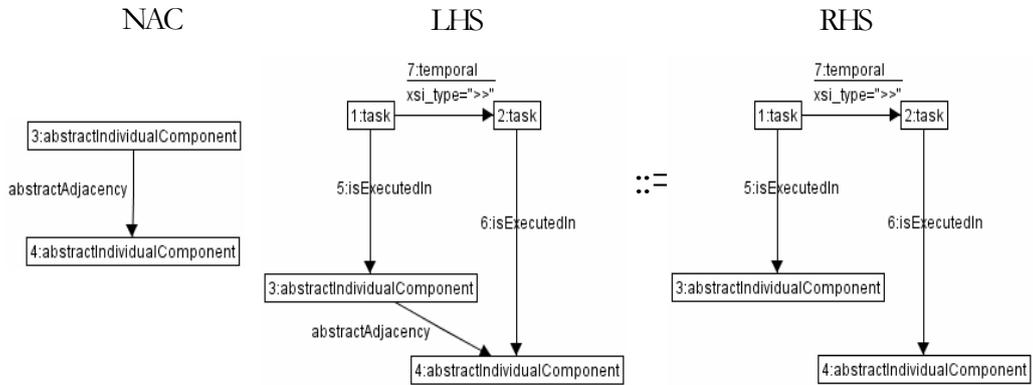


Rule 7-4 Create an AIC for leaf tasks

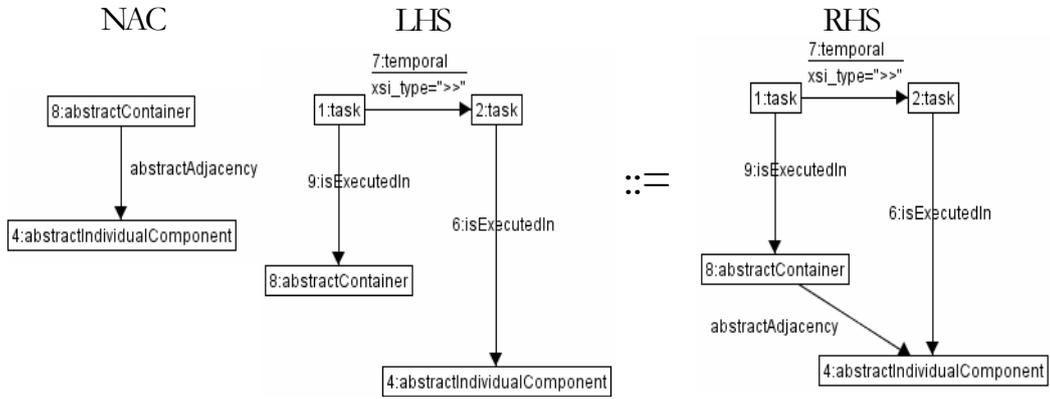


Rule 7-5 Iterative tasks are mapped onto repetitive AC

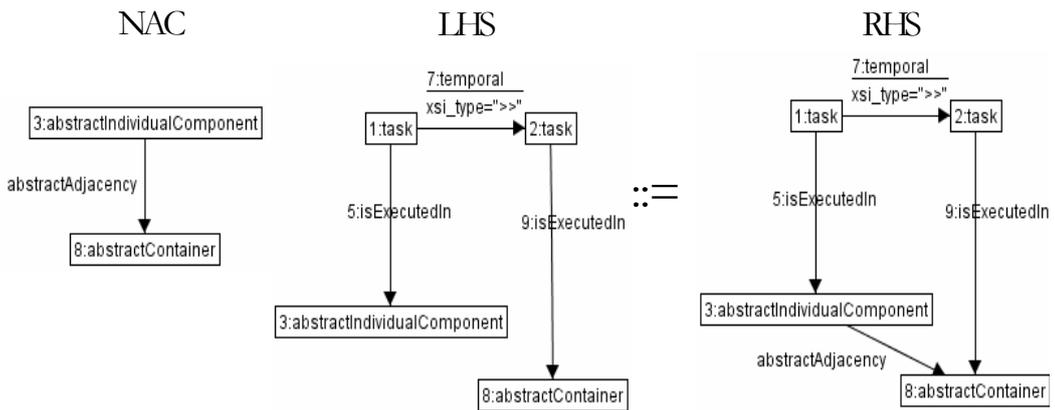
Chapter 7. Validation



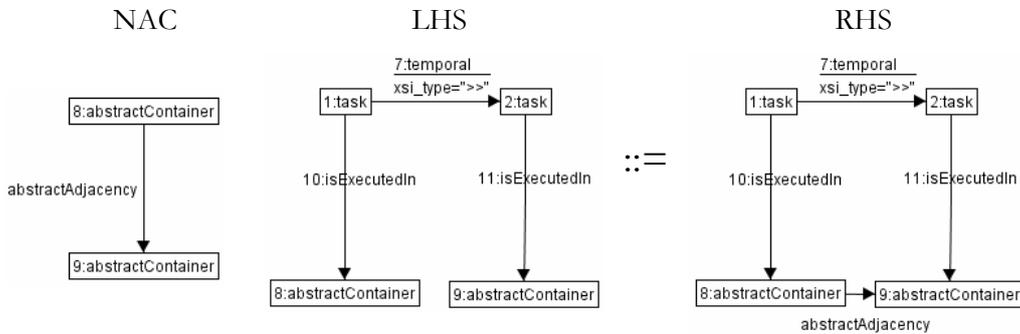
Rule 7-9 Deriving abstract adjacency for <AIC,AIC> couple



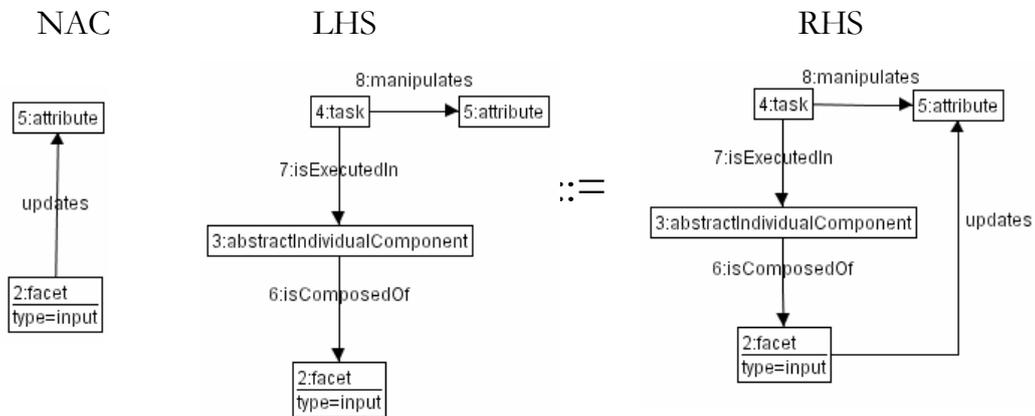
Rule 7-10 Deriving abstract adjacency for <AC,AIC> couple



Rule 7-11 Deriving abstract adjacency for <AIC,AC> couple



Rule 7-12 Deriving abstract adjacency for <AC,AC> couple



Rule 7-13 Derivation of the updates relationship for an input facet

7.1.1.b.2 Selection of AIC

Each AIC can be equipped with facets describing its main purpose/functionality. As explained in Chapter 4, these facets are derived from the combination of the task model, the domain model, and the mappings between them. The mappings between the task and the domain models have been described above. We illustrate some of the rules applicable to the present case study. From these mappings it can be derived that:

- AICs **create name** and **create zipCode** are equipped with an input facet of type “create attribute value”.
- AICs **select sex** and **select ageCategory** are equipped with an input facet of type “select attribute value”. The enumerated values associated to the attribute are transferred as selection value of the facet from the domain model.
- AIC **Show Question** is equipped with an output facet of type “output attribute value” (i.e., the question title).

Chapter 7. Validation

- AIC **Select Answer** is equipped with an input facet of type “select attribute value”. It is set to repetitive since the amount of answers is only known at run-time: no enumerated values are provided nor attribute instances.
- AIC **Send Questionnaire** is equipped with a facet control that references the name of the method on which it is associated, here `sendQuestionnaire`

7.1.1.b.3 Spatio-Temporal arrangement of abstract interaction objects

We apply Rule 7-9, Rule 7-10, Rule 7-11, Rule 7-12. These rules reveal how implementing hierarchical rules in AGG could be repetitive: one rule should be introduced for each possible couple with AC and AIC as elements, that is a total of four rules.

7.1.1.b.4 Definition of abstract dialog control

Rule 7-9 and the like are applied to realize this sub-step. Similarly to the previous step, a rule is defined for each combination of couple with AC and AIC.

7.1.1.b.5 Derivation of AUI to domain mappings

Rule 7-10 is one of the rules applied in this sub-step. Rule 7-15 is another rule that is applicable to our case. It creates an **updates** relationship between the input facet of an AIC and the attribute manipulated by its associated task.

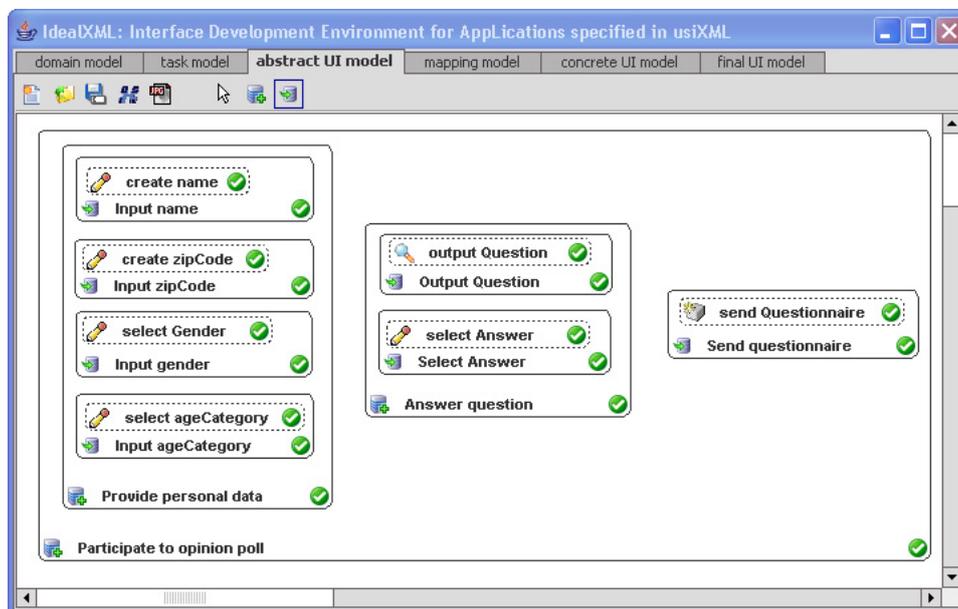


Figure 7-5. IdealXML Mapping from Task and Domain model to Abstract Model A.

7.1.1.b.6 Resulting specification

Following the same mechanism of rule transformation, an abstract individual component (AIC) is created for every leaf task found the task model, insert name, insert zip code, select gender, select age category, show question, select answer and send questionnaire. Each AIC can be equipped with facets describing its main purpose/functionality. These facets are derived from the combination of task model, domain model and the mappings between them. Task definitions have information that is relevant for the mappings, such as: `userAction`, which could be: create, delete, modify, among others. According to these mappings it can be derived that AICs create name and create zipCode are equipped with an input facet of type “create attribute value”. The generated AUI is shown in Figure 7-5.

7.1.1.b.7 Identification of abstract UI structure for AUI “B”

So far the structure of the AUI has been defined; the next step is to specify navigation that might be needed from one container to another. In this context the task model structure plays an important role as well, as it provides information about accessibility of the different elements of the UI. A fragment of the UI might not be enabled, maybe even not visible, when the task that is *executed in* the container requires a previous task to be finished. Then task operators are considered to specify navigation of the UI.

The basic task operator is the task decomposition. In this subsection we illustrate the case when navigation AICs are added to intermediate containers. By intermediate containers we meant not the first, neither the last container of the application, these two containers might just be composed of a forward navigation AIC (the first container) and a backward navigation AIC (the last container). For the intermediate containers forward and backward arrows are needed. The LHS of the transformation rule (Figure 7-6) identify the case when a container is intermediate. The identification criteria search for patterns where three abstract container, notice that neither the first nor the last container will ever meet this pattern.

The RHS rule (Figure 7-7) considers that not just AIC for navigation are needed but also the task tree structure changes as new tasks are added. In this case what is going to happened is that a copy of the original task tree (38: task) is copied to a new dummy task (55:1 task), thus, keeping the original structure of the task tree without change.

Then, new tasks can be associated to the task *55:1 task* without losing consistency of the original task model, particularly, tasks for navigation. Notice that task modelling guidelines are preserved as there is no ambiguity with the task operator.

Chapter 7. Validation

Automatically the user action (*task type*) is assigned with the value corresponding to our canonical list of action types.

The NAC for this transformation rule (Figure 7-8) points to avoid the repetition of the same transformation. Below the NAC in which we avoid the repetition of such transformation when the same pattern of transformation is matched.

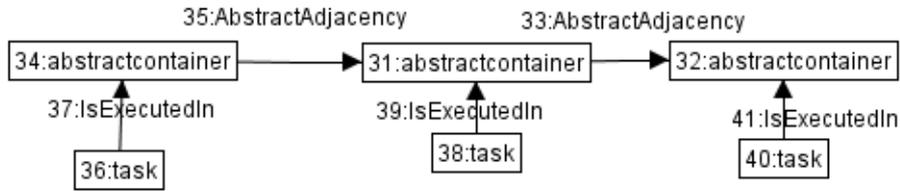


Figure 7-6. LHS Rule for creating navigation facet for abstract containers.

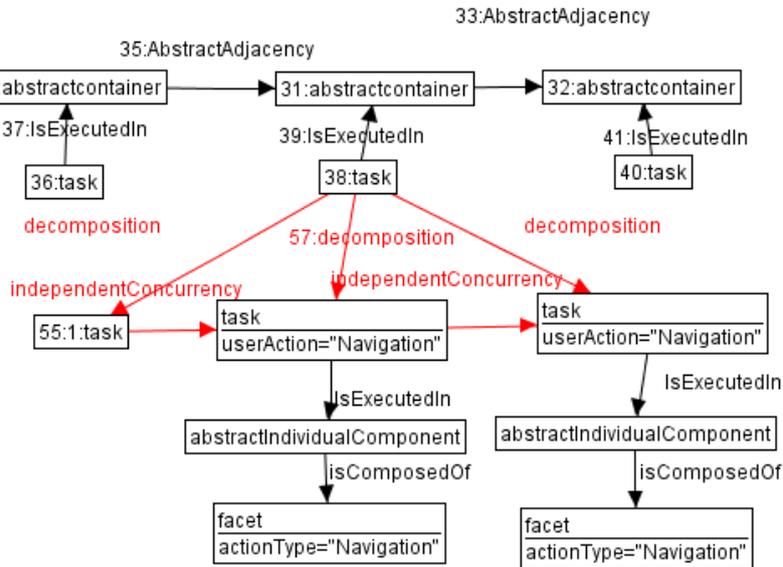


Figure 7-7. RHS Rule for creating navigation facet for abstract containers.

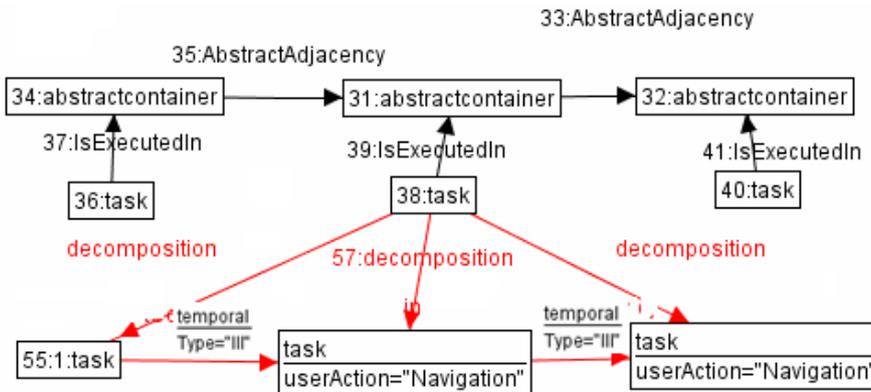


Figure 7-8. NAC for creating navigation facet for abstract containers.

Chapter 7. Validation

Analogously, the transformation rule to add a backward AIC to the last container is described. In this step the last container is identified with the LHS rule (Figure 7-9), just two abstract containers and an adjacency between them is searched. While many others AC could match this rule, even the first container, then more information is needed to differentiate the containers. Those rules are expressed in term of NACs. More clearly, the LHS rule below shows a requirement to identify either the first or the last abstract container in a task tree. When an abstract container (2:abstractcontainer) with abstract adjacency (5:abstractAdjacency) to another abstract container (3:abstractcontainer) an related with an abstract containment (7:abstractContaiment) with an upper level abstract container.

This initial state can be then compared to the left sibling to check (Rule on the left in Figure 7-10) if there are any abstract containers related to the abstract container (2:abstractcontainer). A second NAC required is to check if the abstract container (1:abstractcontainer) is the root of abstract containers (Rule on the right in Figure 7-10).

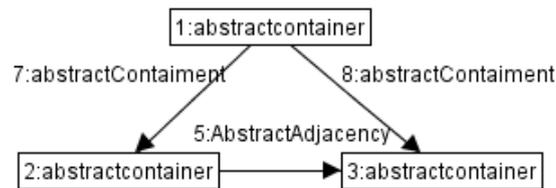


Figure 7-9. LHS Rule for creating navigation facet for the first abstract container.

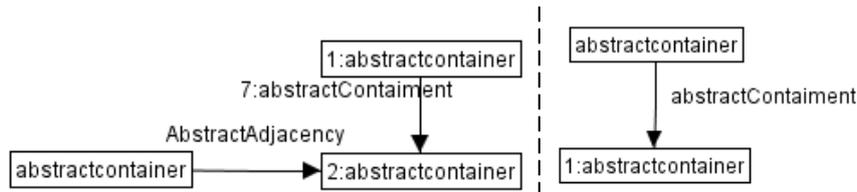


Figure 7-10. NACs for creating navigation facet for the first abstract containers.

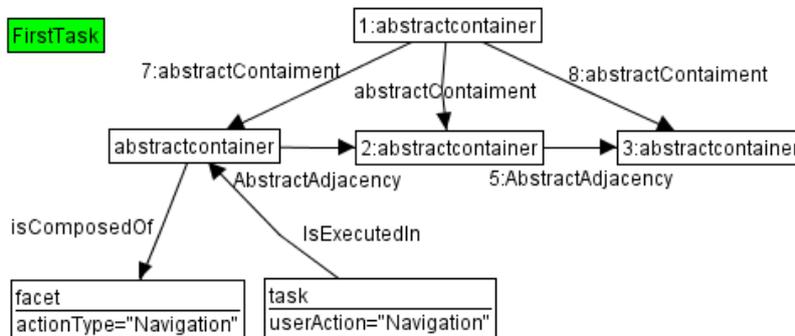


Figure 7-11. RHS Rule for creating navigation facet for the first abstract container.

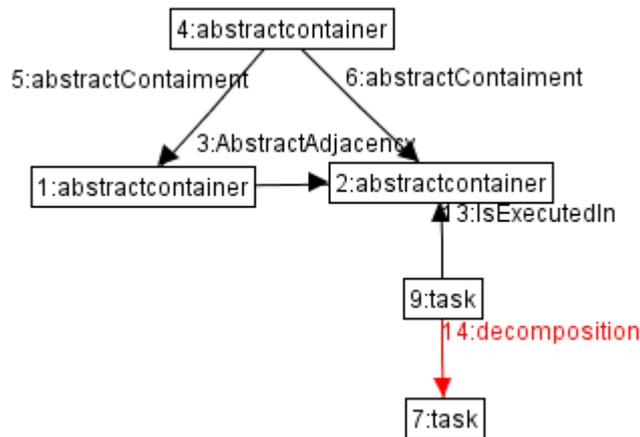


Figure 7-12. LHS Rule for creating navigation facet for the last abstract container.

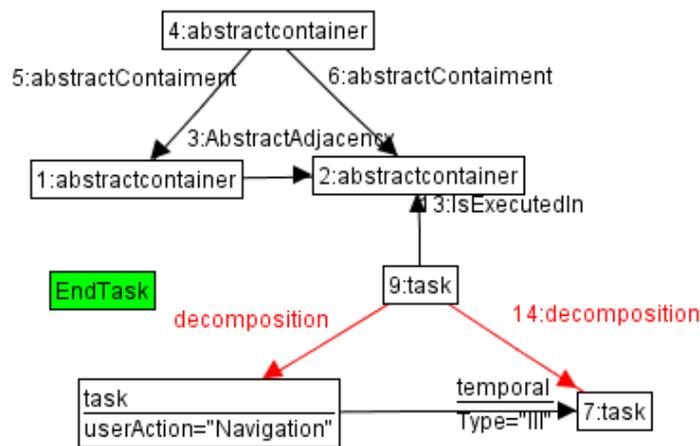


Figure 7-13. RHS Rule for creating navigation facet for the last abstract container.

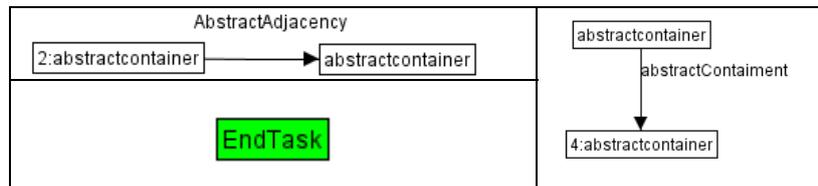


Figure 7-14. NAC for creating navigation facet for the last abstract containers.

The RHS rule that generates the navigation component is similar to the rule previously used as follows. Notice that the creation of this rule requires the creation of a new NAC, if not, infinitely the system could generate abstract containers for the new task created. Flags can be generated for this purpose, dummy nodes that are not related to the system but just to control the flow can be added and/or re-

Chapter 7. Validation

moved. So the appropriate RHS is below. Finally the new NAC required is **FirstTask**.

Analogous to the first task, to create navigation for the last task, we will need the almost the same LHS rule (Figure 7-12), but some differences are required, such as to identify the task that is executed in the abstract container (2:abstractcontainer) to connect it to the new navigation task.

The RHS rule (Figure 7-13) applied to the above LHS rule, we assume in this case that the last task is decomposed in at least two other subtasks, and we are just interested in linking our new navigation task to any sub-task (9:task). The three NAC are listed below (Figure 7-14). The generated abstract user interface is shown in Figure 7-15.

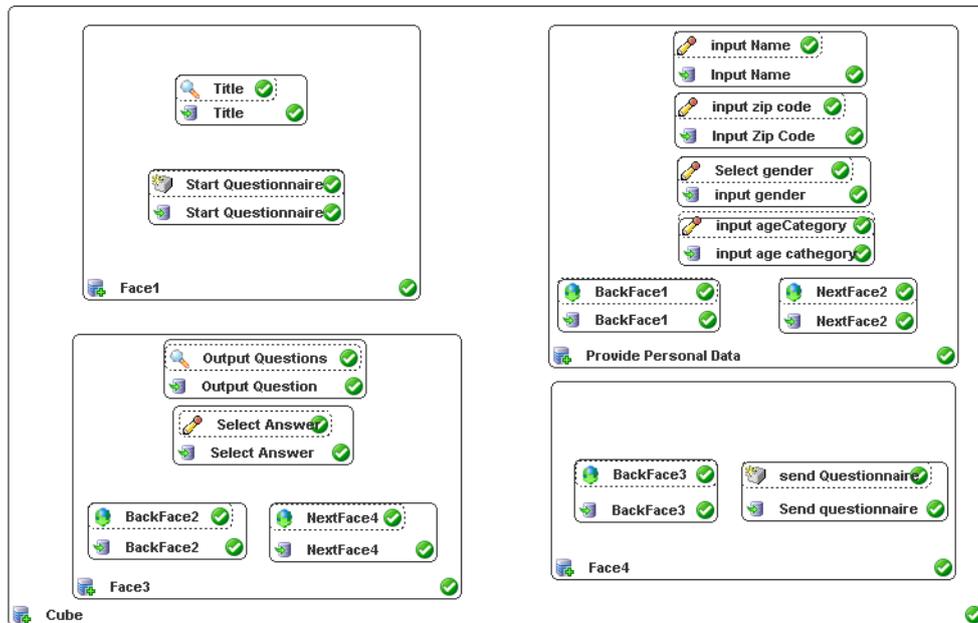


Figure 7-15. IdealXML Mapping from Task and Domain model to Abstract Model B.

The corresponding UsiXML specifications, generated in IdealXML, correspond to the AIO decomposition.

```
<abstractContainer id="idao0" name="Cube" splittability="true">
<abstractContainer id="idao1" name="Face1">
  <abstractIndividualComponent id="idao5" name="Title">
    <output id="idao6" name="Title" outputContent="Welcome to the Virtual Polling system" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao7" name="Start Questionnaire">
    <control id="idao8" name="Start Questionnaire" actionType="interaction"
      event="startQuestionnaire" />
  </abstractIndividualComponent>
</abstractContainer>
```

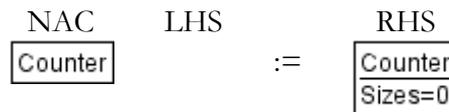
```
</abstractContainer>
<abstractContainer id="idao2" name="Provide Personal Data">
  <abstractIndividualComponent id="idao9" name="Input Zip Code">
    <input id="idao15" name="input zip code" actionType="interaction" dataType="String"
      attributeDomainCharacterization="zipCode" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao10" name="Input Name">
    <input id="idao14" name="input Name" actionType="interaction" dataType="String"
      attributeDomainCharacterization="name" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao11" name="input gender">
    <input id="idao16" name="Select gender" actionType="interaction" dataType="String"
      attributeDomainCharacterization="gender" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao12" name="input age cathegory">
    <input id="idao17" name="input ageCategory" actionType="interaction" dataType="String"
      attributeDomainCharacterization="ageCategory" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao13" name="BackFace1">
    <navigation id="idao18" name="BackFace1" actionType="interaction" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao19" name="NextFace2">
    <navigation id="idao20" name="NextFace2" actionType="interaction" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao3" name="Face3">
  <abstractIndividualComponent id="idao26" name="Output Question">
    <output id="idao29" name="Output Questions" actionType="interaction"
      outputContent="Questions" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao27" name="Select Answer">
    <input id="idao28" name="Select Answer" actionType="interaction" dataType="String"
      attributeDomainCharacterization="answer" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao31" name="BackFace2">
    <navigation id="idao34" name="BackFace2" actionType="interaction" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao32" name="NextFace4">
    <navigation id="idao33" name="NextFace4" actionType="interaction" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao4" name="Face4">
  <abstractIndividualComponent id="idao22" name="Send questionnaire">
    <control id="idao23" name="send Questionnaire" actionType="interaction"
      event="sendQuestionnaire" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao24" name="BackFace3">
    <navigation id="idao25" name="BackFace3" actionType="interaction" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
```

7.1.1.c Step 3: From Abstract model to Concrete User Interface model

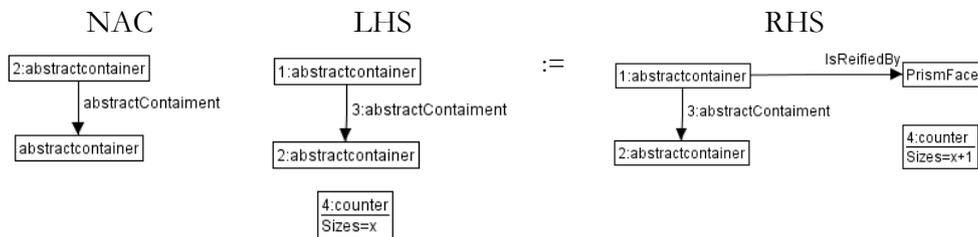
The third step implies a transformational system that is composed of necessary rules for realizing the transition from AUI to CUIs. For this purpose, other design rules could be encoded in UsiXML so as to transform the AUI into different CUI. Since the AUI model is a CIM, it is supposed to remain independent of any im-

Chapter 7. Validation

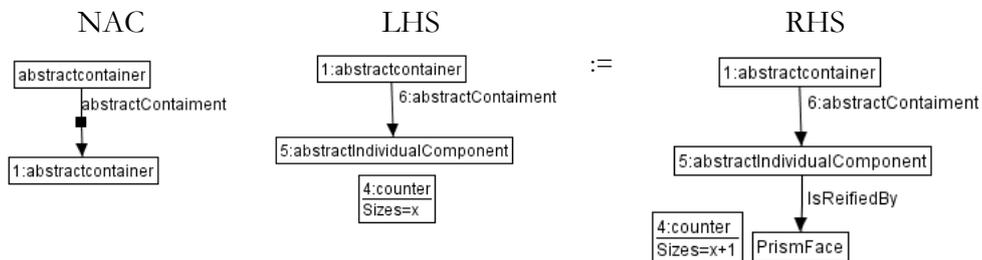
plementation. However, when it comes to transform this AUI into a corresponding CUI or several variants of it, platform concerns come into consideration. For this purpose, several design rules [Limb04c, Stan08] exist that transform the AUI into CUIs with different design options that will then be turned into final code when generated. Our concern is to expand those in order to support the concretization of the UI into a 3DUI.



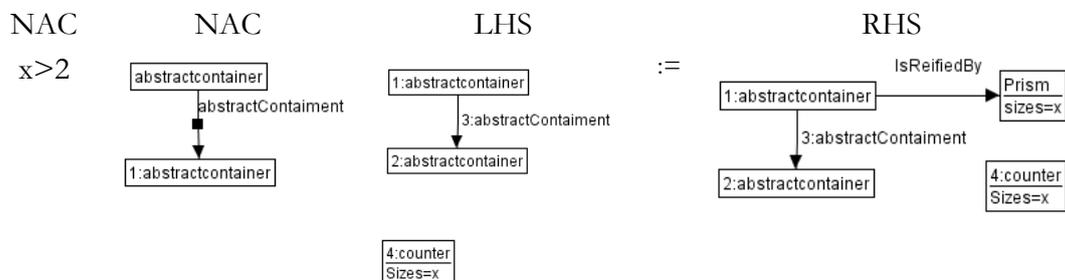
Rule 7-14 Derivation of the updates relationship for an input facet



Rule 7-15 A creation of Prism face derived from containment relationships at the abstract level



Rule 7-16 A creation of Prism face derived from containment relationships at the abstract level



Rule 7-17 A creation of Prism face derived from containment relationships

7.1.1.c.1 Reification of AC into CC

The simple heuristic in 2D solving the current problem consists of representing all tasks into one single window. For the 3DUI we might have such concretization but for this example we chose to have the prism as a container. Each AC becomes a prism face, not including the top and the bottom if the prism. Then any surface but the top level could become the virtual area where widgets can be attached (used for the variant in which objects will be floating in the space) or any other container such a prisms (the number of sides for the prisms would be based on the sublevel of containers required). Each abstract container at level “leaf-1” is transformed into a prism face. The main container is mapped to a prism based on a square, in this case a predefined cube (accordingly to the fact that at least four containers are required). This could be change in run time, as we will show but this consequently will increase the amount of prim faces and as a consequence the polygon-based number of sizes. First, we create the rule 7-14 to create the counter for the prism sizes. If the is no counter, create it an initialize it to 0.

Abstract Interaction Component	Facet Specification	Information to take into account	Possible Concrete Interaction Component
“start”	Control	Feedback	A trigger
“create name” and “create zipCode”	Create attribute value	Data type, domain characteristics	A text output with a text input associated to it
“select gender and select ageCategory”	Select attribute value + selection values known	Data type, domain characteristics, selection values	A dropdown list , a group of radio buttons textual or characters.
“Show Questionnaire”	Output (value unknown)	Attribute, data type, domain characteristics	An output text
“Select Answer”	Select attribute value + repetitive (selection values not known)	Data type, domain characteristics	A dropdown list, a group of option buttons
“Send Questionnaire”	Control	Feedback	A trigger
“Navigation”	Navigation	Feedback	A trigger

Table 7-1. Correspondence between AIO types and CIO types.

Then we mapped to a prism face each container at level “leaf-1” with rule 7-15. We count also the number of AC to determine the number of sizes of the prism. There is also a need to add a similar rule but for AIC which, as in our example the first task to start the application, will be in a last face but is executed in an AIC

and not in a AC. The rule 7-16 solves this problem. Finally we create a prism with x sizes (Figure 7-17).

7.1.1.c.2 Selection of CICs

This sub-step involves the highest number of rules of all transformation sets as the different combinations of facet types, data types, cardinalities,..., are numerous. Table 7-1 provides the subset of rules applied in this case study. The designer can choose among the different alternatives provided by these rules.

7.1.1.c.3 CIC placement

Physical constraints related to the size of the container and 3DCIC are considered. Each face of the cube could have just as much a pair of CIC at the same level; consequently this depends on the size of the components. In the case of text component this also depends on the size of the string that will display.

7.1.1.c.4 Navigation definition

Navigation specifies how the visibility property of CCs is set and, consequently, defines transitions between them. Since all elements are not presented simultaneously into the same prism face, there is a particular need to define a sophisticated navigation scheme. Some schemes can be added for this purpose, so we identify that for any prism there is a need to add triggers that allows navigation, i.e. go back and forward to each face of the prism (this will be restricted depending on the navigation info from the task model, maybe certain information could not be accessible while other is accessed). In this particular case just the fourth faces allows navigation.

7.1.1.c.5 Resulting specification

The resulting specifications are obtained by realizing the above development sub-steps. Figure 7-16 presents a mock-up of the graphical UI. For the section start one face of the cube will launch this task, the second face corresponds to the *provide personal data* task. In this part the rectangles next to the name and zip code, corresponds to the surface zone that will render the input text. The arrows for navigation are shown in the bottom of faces two to four. The cube as a container is divided in 6 faces but the top and the bottom are useless in this case. Each of the fourth faces of the fourth sized prism (cube) has a purpose that is to render each of the 3D graphical individual components (TDGIO). As four AC were required the cube clearly works as an option to render each AC in each of its four faces. Later we will show another option to render the same problem so as to increment the quantity of question to use another shape instead of the cube.

Chapter 7. Validation

XML	
▲ Cube	
id	C1
name	CubePole
defaultContent	Virtual Polling System
size	2.0, 2.0, 2.0
solid	true
isVisible	true
isEnabled	true
▲ Group	
▼ CubeFace id=C1	
▼ CubeFace id=C2	
▼ CubeFace id=C3	
▼ CubeFace id=C4	

The UsiXML resulting from this process is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<Cube id="C1" name="CubePole" defaultContent="Virtual Polling System" size="2.0, 2.0, 2.0"
solid="true" isVisible="true" isEnabled="true">
  <Group>
    <CubeFace id="C1">
      <SphereTrigger defaultContent="Start" radius="1.5" solid="True" isVisible="true"
isEnabled="true">
        <Transform scale="8.23 8.23 8.23" translation="0.27 12.14 18.30"/>
        <TouchSensor id="TS1" enabled="True"/>
        <Appearance name="ButtonAppe" id="App1">
          <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
      </SphereTrigger>
    </CubeFace>
    <CubeFace id="C2">
      <Outputtext3D defaultContent="Name" id="T1">
        <Appearance name="ArrowAppe3" id="Back3">
          <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
        <Transform translation="-1.51 -0.11 0.19"/>
      </Outputtext3D>
      <Inputtext3D defaultContent="" id="IT1">
        <Transform translation="0.025 -0.11 0.19"/>
        <Appearance name="ArrowAppe3" id="Back3">
          <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
        <TouchSensor id="TST1" enabled="True"/>
      </Inputtext3D>
      <Outputtext3D defaultContent="Zip Code" id="T2">
        <Transform translation="-1.39 -0.22 0.02"/>
        <Appearance name="ArrowAppe3" id="Back3">
          <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
      </Outputtext3D>
      <Inputtext3D defaultContent="" id="IT1">
        <Transform translation="0.09 -0.22 0.02"/>
        <Appearance name="ArrowAppe3" id="Back3">
```

Chapter 7. Validation

```
<Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
    emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TST2" enabled="True"/>
</Inputtext3D>
<Outputtext3D defaultContent="Gender" id="T3">
  <Transform translation="-1.60 -0.33 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
        emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</Outputtext3D>
<radioButton id="" defaultContent="M" solid="True" isVisible="true" isEnabled="true"
    defaultState="True" groupName="Gender"/>
<Transform translation="-.60 -0.33 0.0"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB1" enabled="True"/>
<radioButton/>
<radioButton id="" defaultContent="F" solid="True" isVisible="true" isEnabled="true"
    defaultState="False" groupName="Gender"/>
<Transform translation="0.0 -0.33 0.0"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB2" enabled="True"/>
<radioButton/>
<Outputtext3D defaultContent="Age" id="T4">
  <Transform translation="-1.6 -0.44 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
        emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</Outputtext3D>
<radioButton id="" defaultContent="18-25" solid="True" isVisible="true" isEnabled="true"
    defaultState="True" groupName="Age"/>
<Transform translation="-.60 -0.44 0.0"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB1" enabled="True"/>
<radioButton/>
<radioButton id="" defaultContent="25-45" solid="True" isVisible="true" isEnabled="true"
    defaultState="False" groupName="Age"/>
<Transform translation="0.0 -0.44 0.0"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB1" enabled="True"/>
<radioButton/>
<radioButton id="" defaultContent="45+" solid="True" isVisible="true" isEnabled="true"
    defaultState="False" groupName="Age"/>
<Transform translation="0.6 -0.44 0.0"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
```

Chapter 7. Validation

```
        emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
    </Appearance>
    <TouchSensor id="TSRB1" enabled="True"/>
    <radioButton/>
    <ArrowTrigger id="A1" defaultContent="Back" solid="True" isVisible="true"
        isEnabled="true">
        <Transform scale="0.76 0.18 1.0" translation="17.3 5.9 7.08" rotation="0.0 1.0 0.0
            1.570796"/>
        <TouchSensor id="TS2" enabled="True"/>
        <Appearance name="ArrowAppe1" id="Back1">
            <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
    </ArrowTrigger>
    <ArrowTrigger id="A2" defaultContent="Next" solid="True" isVisible="true"
        isEnabled="true">
        <Transform scale="8.23 8.23 8.23" translation="-18.6 5.4 7.5"/>
        <TouchSensor id="TS3" enabled="True"/>
        <Appearance name="ArrowAppe2" id="App2">
            <Material diffuseColor="0.8 0.0 0.0" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
    </ArrowTrigger>
</CubeFace>
<CubeFace id="C3">
    <Outputtext3D defaultContent="The professor teach what it was expected?" id="T4">
        <Transform translation="-0.62 0.0 -0.52"/>
        <Appearance name="TEXT" id="TEXT">
            <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
    </Outputtext3D>
    <radioButton id="" defaultContent="Yes" solid="True" isVisible="true" isEnabled="true"
        defaultState="True" groupName="YesNo1"/>
        <Transform translation="0.0 -0.5 0.0"/>
        <Appearance name="TEXT" id="TEXT">
            <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
        <TouchSensor id="TSRB1" enabled="True"/>
    </radioButton>
    <radioButton id="" defaultContent="No" solid="True" isVisible="true" isEnabled="true"
        defaultState="False" groupName="YesNo1"/>
        <Transform translation="0.5 -0.5 0.0"/>
        <Appearance name="TEXT" id="TEXT">
            <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
        <TouchSensor id="TSRB2" enabled="True"/>
    </radioButton>
    <Outputtext3D defaultContent="Did you like the course?" id="T4">
        <Transform translation="-0.62 -1.0 -0.52"/>
        <Appearance name="TEXT" id="TEXT">
            <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
                emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
    </Outputtext3D>
    <radioButton id="" defaultContent="Yes" solid="True" isVisible="true" isEnabled="true"
        defaultState="True" groupName="YesNo2"/>
    <Transform translation="0.0 -1.5 -0.52"/>
```

Chapter 7. Validation

```
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
    emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB3" enabled="True"/>
<radioButton/>
<radioButton id="" defaultContent="No" solid="True" isVisible="true" isEnabled="true"
  defaultState="False" groupName="YesNo2"/>
<Transform translation="0.5 -1.5 -0.52"/>
<Appearance name="TEXT" id="TEXT">
  <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
    emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
</Appearance>
<TouchSensor id="TSRB4" enabled="True"/>
<radioButton/>
<ArrowTrigger id="A3" defaultContent="Back" solid="True" isVisible="true"
  isEnabled="true">
  <Transform scale="0.76 0.18 1.0" translation="3.46 5.27 -19.06" rotation="0.0 1.0
    0.0 1.570796"/>
  <TouchSensor id="TS4" enabled="True"/>
  <Appearance name="ArrowAppe2" id="Back2">
    <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</ArrowTrigger>
<ArrowTrigger id="A4" defaultContent="Next" solid="True" isVisible="true"
  isEnabled="true">
  <Transform scale="8.23 8.23 8.23" translation="-18.6 5.4 7.5"/>
  <TouchSensor id="TS5" enabled="True"/>
  <Appearance name="ArrowAppe2" id="App2">
    <Material diffuseColor="0.8 0.0 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</ArrowTrigger>
</CubeFace>
<CubeFace id="C4">
  <Outputtext3D defaultContent="Did you enjoy the course?" id="T4">
    <Transform translation="-1.60 -0.33 0.0"/>
    <Appearance name="TEXT" id="TEXT">
      <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
        emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
    </Appearance>
  </Outputtext3D>
  <radioButton id="" defaultContent="Yes" solid="True" isVisible="true" isEnabled="true"
    defaultState="True" groupName="YesNo3"/>
  <Transform translation="-0.60 -0.33 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TSRB5" enabled="True"/>
  <radioButton/>
  <radioButton id="" defaultContent="No" solid="True" isVisible="true" isEnabled="true"
    defaultState="False" groupName="YesNo3"/>
  <Transform translation="0.0 -0.33 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TSRB6" enabled="True"/>
</CubeFace>
```

```

</radioButton/>
<ArrowTrigger id="A5" defaultContent="Back" solid="True" isVisible="true"
  isEnabled="true">
  <Transform scale="0.76 0.18 1.0" translation="-19.15 5.3 -6.5" rotation="0.0 1.0 0.0
    1.570796"/>
  <TouchSensor id="TS6" enabled="True"/>
  <Appearance name="ArrowAppe3" id="Back3">
    <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</ArrowTrigger>
<SphereTrigger id="B2" defaultContent="Send" radius="1.5" solid="True"
  isVisible="true" isEnabled="true">
  <Transform scale="8.23 8.23 8.23" translation="-18.6 5.4 7.5"/>
  <TouchSensor id="TS7" enabled="True"/>
  <Appearance name="ButtonAppe2" id="App2">
    <Material diffuseColor="0.8 0.0 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</SphereTrigger>
</CubeFace>
</Group>
</Cube>

```

How each face could look in a 2D view is shown below. The arrows show the need for navigation.

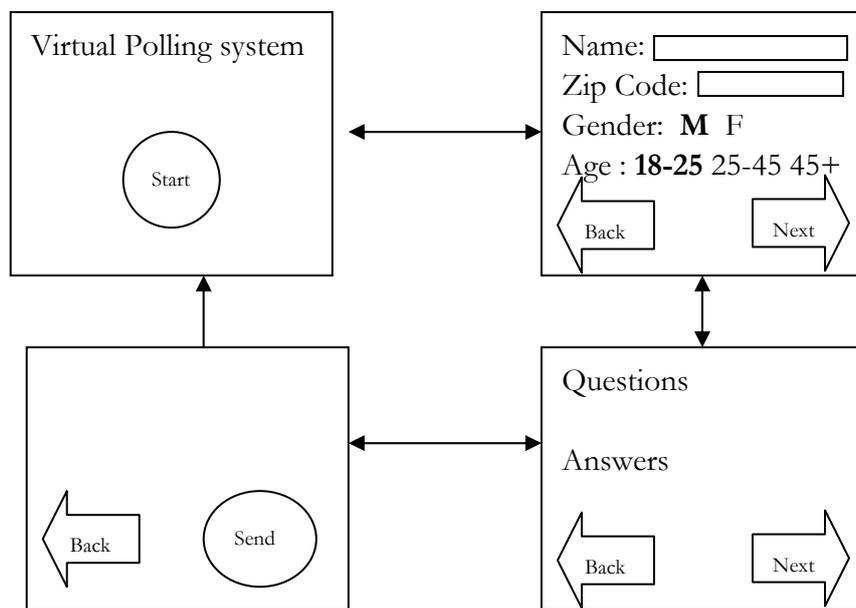


Figure 7-16. Mock-up of the UI.

7.1.1.d Step 4: From Concrete model to Final User Interface

The fourth step involves the transformation from CUI to FUI. In screenshot of Figure 7-17, the decomposition of faces is fine-grained: the information related to the person are first acquired in a rotating cube (which was selected as the con-

Chapter 7. Validation

tainer), then each pair of questions is presented at a time with the facilities of going forward or backward like a wizard (using arrow triggers). Since only 3 questions and one set of person information are considered, a cube is selected to present each of the fourth part. If for any reason, more questions were defined, let us say 5, a regular volume with 6 faces would be generated instead.

Each face of the cube is mapped to each of the faces. Below we show the UsiXML code generated for the CUI, corresponding to the first AC, that just have one AIC, which is mapped to a Sphere Trigger. The cube as the principal container has its title, the attribute *defaultContent* (inherited from CIO model). This title is part of the first face of the cube. The second cube attribute is its size.

```
<Cube id="C1" name="CubePole" defaultContent="Virtual Polling System" size="2.0, 2.0, 2.0"
  solid="true" isVisible="true" isEnabled="true">
  <Group>
    <CubeFace id="C1">
      <SphereTrigger defaultContent="Start" radius="1.5" solid="True" isVisible="true"
        isEnabled="true">
        <Transform scale="8.23 8.23 8.23" translation="0.27 12.14 18.30"/>
        <TouchSensor id="TS1" enabled="True"/>
        <Appearance name="ButtonAppe" id="App1">
          <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
            emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
        </Appearance>
      </SphereTrigger>
    </CubeFace>
    <CubeFace id="C2">
    <CubeFace id="C3">
    <CubeFace id="C4">
  </Group>
</Cube>
```

As in this example we have three different groups of CIO, which are:

1. A text output and a text input for the create name and zip code tasks.
2. A text output and a group of radio buttons for the select gender and age category tasks.
3. Arrow triggers for navigation.

We show the CUI UsiXML code related to each of the above two situations but showing just one of its two cases. Below the section that corresponds to the name creation. OutputText (specialized in any kind of output text) and InputText (specialized in any kind of input text) are two components from the TDGIC. For an InputText there is a need to declare an event sensor to listen to events that could be triggered by the InputText. Such events could be, is Over (the pointer is on the input text), key Down (to identify the key pressed from the keyboard). The rest of the code describes the appearance for the text and its position (transform).

Chapter 7. Validation

```
<Outputtext3D defaultContent="Name" id="T1">
  <Appearance name="ArrowAppe3" id="Back3">
    <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <Transform translation="-1.51 -0.11 0.19"/>
</Outputtext3D>
<Inputtext3D defaultContent="" id="IT1">
  <Transform translation="0.025 -0.11 0.19"/>
  <Appearance name="ArrowAppe3" id="Back3">
    <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TST1" enabled="True"/>
</Inputtext3D>
```

The second part of the face shows an output text followed by a radio buttons group. Below the code related to the radio buttons group. We use the same attributes that are described in UsiXML for this component, as in 3D there is no difference. We need a radio group name, a default state (whether is selected or not). The rest of the code describes the appearance for the text and its position (transform).

```
<radioButton id="" defaultContent="Yes" solid="True" isVisible="true" isEnabled="true"
  defaultState="True" groupName="YesNo1"/>
  <Transform translation="0.0 -0.5 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TSRB1" enabled="True"/>
</radioButton/>
<radioButton id="" defaultContent="No" solid="True" isVisible="true" isEnabled="true"
  defaultState="False" groupName="YesNo1"/>
  <Transform translation="0.5 -0.5 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TSRB2" enabled="True"/>
</radioButton/>
```

Finally the arrow triggers used for navigation purposes. The default content of the trigger defines the text attached to them. The rest of the code describes the appearance and its position (transform). There is a need for a sensor to trigger an action, in this case will be the translation of the cube +- 90 digress related to Y axis.

```
<ArrowTrigger id="A1" defaultContent="Back" solid="True" isVisible="true"
  isEnabled="true">
  <Transform scale="0.76 0.18 1.0" translation="17.3 5.9 7.08" rotation="0.0 1.0 0.0
    1.570796"/>
  <TouchSensor id="TS2" enabled="True"/>
  <Appearance name="ArrowAppe1" id="Back1">
    <Material diffuseColor="0.8 0.8 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
```

```
</Appearance>
</ArrowTrigger>
<ArrowTrigger id="A2" defaultContent="Next" solid="True" isVisible="true"
  isEnabled="true">
  <Transform scale="8.23 8.23 8.23" translation="-18.6 5.4 7.5"/>
  <TouchSensor id="TS3" enabled="True"/>
  <Appearance name="ArrowAppe2" id="App2">
    <Material diffuseColor="0.8 0.0 0.0" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
</ArrowTrigger>
```

The third and four faces for the cube, do not have any different object that could be interesting to describe, just output text, radio buttons groups and a Sphere trigger. What is relevant is just that the mapping to each FUI, in this case in VRML corresponds to the CUI model, describe in UsiXML. If the set of questions could be more extended, automatically the sides of the prism are expanded. In the example above, the cube fits the necessities of a pool of questions. In Figure 7-18 a 6-sizes prism serves as container instead of the cube. The path through obtain the six-sizes shape is analogous to the previous one, the difference is the quantity of questions at the when generating the final user interface. In this second scenario, instead of four containers there is a need for 2 more, as four more questions were added to the pool. As a consequence the shape required to handle this information varies. The natural size of such presentation might be a prism no bigger than 10-sizes. After that a perspective wall would be the best concretization of the container.

7.1.1.e Reconstruction of the case study in Java 3D

The scenario proposed in the previous example uses containers to render the information that will be shown in the virtual space. Normally controls and any CIO of the UI in 2D are attached to any kind of container. In virtual space the counter part of the window is the virtual space itself. So, object could be rendered in the virtual space, floating, without any need of containers.

Designers are allowed to decide whether they prefer to use containers (as we did in the face of the prism) or attach directly the components to the virtual space. This design decision should be taken when passing from the Abstract model to the concrete model. We could decide whether container contained in the main container (the virtual space in 3D) will be attached to a surface that then will corresponds to a prism or will be attached directly to the virtual space. The second option will be used to show the results of this kind of decision. If we do not mapped the containers to surfaces then there will not be a final instance of any kind of prism to render each surface that serves as container. The virtual space will render all the content of the container describe in the CUI model.

In Figure 7-19, the screenshot reproduces the worlds generated in Java3D where each AC (provision personal data and answer question) is mapped onto the virtual space. All AICs belonging to each AC are then mapped recursively onto Java3D widgets depending on their data type. In this particular case, the designer selected also the graphical representation if any, along with the textual representing. In this visualization, we propose another way to represent the category selection. Instead of using a comboBox, or the traditional view of icons attached to radio button, we proposed the use of 3D personages instead of icons. This 3D graphic representation of the option could reinforce the understanding, notice that we keep the text below the personages. To obtain this result there is a need to have a look at the CUI model. Each TDCIO has an attribute called icon. This icon in 2D generally corresponds to a bitmap image that is attached to the controls in a UI. In 3D we propose that icons could be any shape, so in this case we use the same definition as in the previous example to define radio button but adding the url corresponding to each radio button. The code could be seen as follows:

```
<radioButton id="" defaultContent="18-35" solid="True" isVisible="true" isEnabled="true"
  defaultState="True" groupName="YesNo1" icon="youngMan.java"/>
  <Transform translation="0.0 -0.5 0.0"/>
  <Appearance name="TEXT" id="TEXT">
    <Material diffuseColor="0.3 0.3 0.3" specularColor="0.11 0.11 0.11"
      emissiveColor="0.0 0.0 0.0" shininess="0.3"/>
  </Appearance>
  <TouchSensor id="TSRB1" enabled="True"/>
</radioButton/>
```

7.1.1.f Reconstruction of the case study for the 3D rendering of its corresponding 2DUI

The UsiXML specifications at the CUI could also be interpreted in VUIToolkit, a rendering engine for 3DUIs specified in UsiXML in VRML97/X3D. In the screenshot of the Figure 7-20, we show the result of using the Toolkit that generates the 3D rendering of how our polling system could look in a 2D user interface. The 2D components have been enriched with volumes. One can discuss that the components are rendered as 3D widgets in a way that remains similar to the “Look & Feel” of 2D widgets, except that the “Feel” is a genuine 3D behaviour. According to this view, this kind of FUI can be interpreted only as a 3D rendering of 2D UIs, even if their specifications are toolkit-independent [Moli05]. This approach provides an option to the use of Java applets UIs to manipulate virtual applications in the Web, instead, the use of the VUIToolkit would not disrupt the 3D “look”.

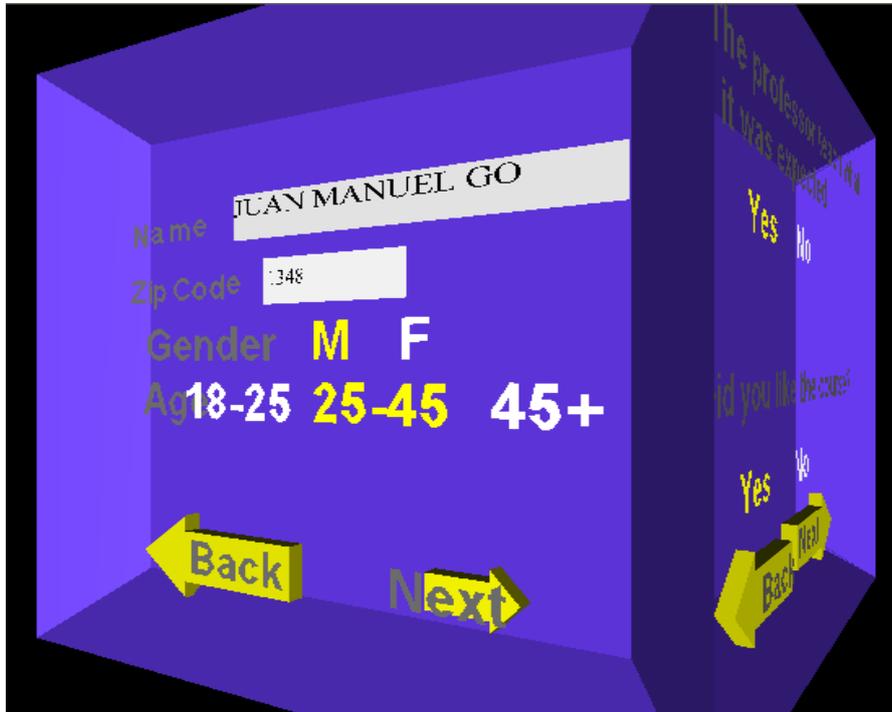


Figure 7-17. Polling system rendered in VRML.

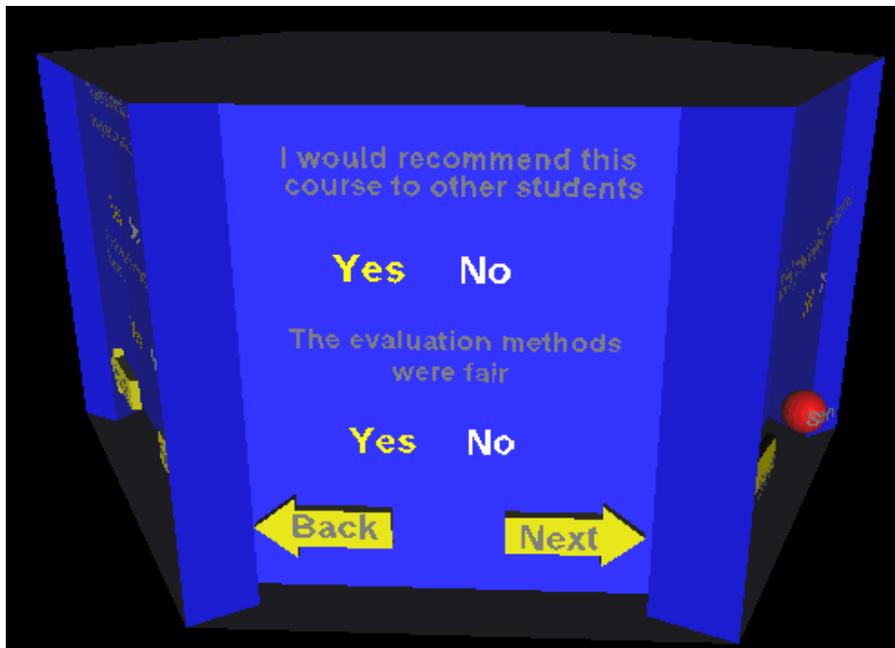


Figure 7-18. Hexahedron Polling System.

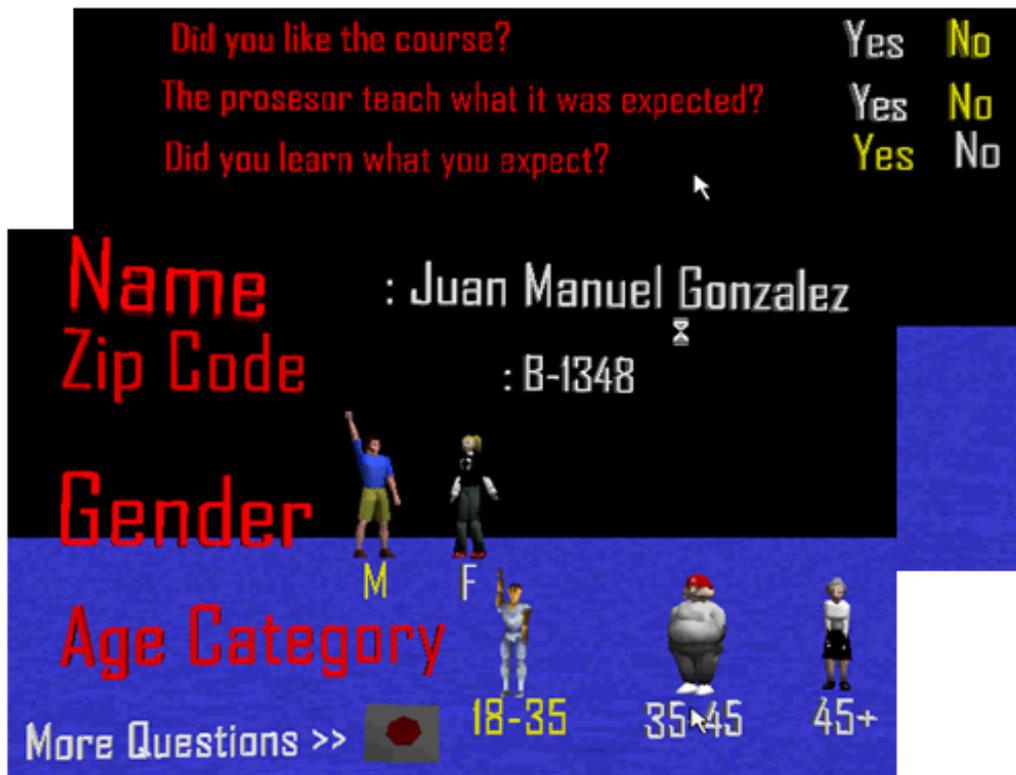


Figure 7-19. Edition of the 3DUI in Maya.

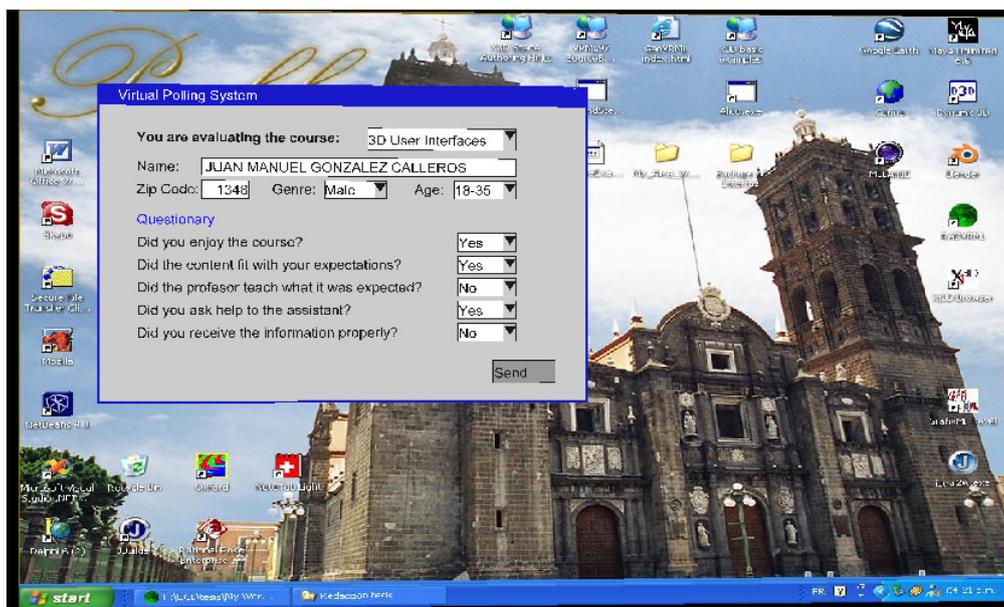


Figure 7-20. Three-D rendering of the 2D interface for the polling system.

7.1.2 Case Study 2: The Student- Trainer Application

In this section we use the student/trainer system case study to illustrate our method. The following scenario illustrates the problems and the need for such a toolkit: due to beginning of the new academic year an education institute specialized in courses for people who is working needs to calculate the number of student per trainer for the each course accordingly to different variables: number of available trainers, salary per trainer, trainer working days, annual working days for students, students salary average and number of students. So far the school has been using an excel spreadsheet however manipulating variables imply some usability disadvantages, boundaries for the variables are not visually available, also, manipulate the value of a variable has two steps type the value then press enter, any other change has to follow the same procedure. They would prefer to manipulate variables using a dedicated UI more interactive were they can select the value of the variables in a flexible way, such is the case is they were using sliders.

7.1.2.a Step 1: The Task and Domain Models

The task model, the domain model, and the mappings between, are all graphically described using IdealXML tool [Mont05] (Figure 7-21). The task tree, depicted using CTT notation, shows the envisioned system. The root task consists of student/trainer is decomposed in two task, one for calculating student per trainer and the second task to exit the system at any time. Calculating student/trainer is divided in two tasks; one where the user selects the variables and the second is a system task where the result is calculated and shown to the user. The user can change any variable iteratively but any change is transferred automatically to the system task to update the result.

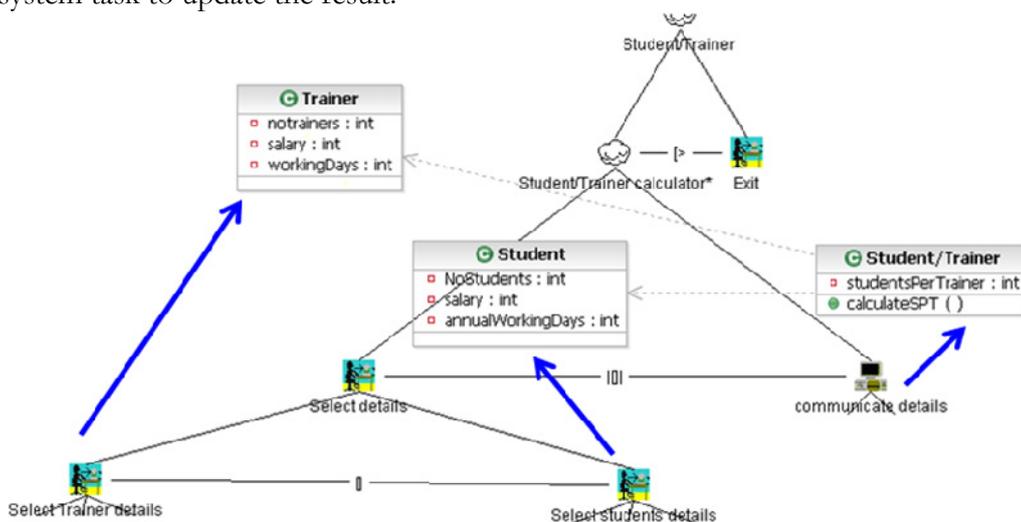


Figure 7-21. Task & Concepts of the student/trainer system.

Chapter 7. Validation

In order to expand the task model and to make the manipulation of the attributes for: trainers (*notrainers*, *salary*, *workingdays*), student (*NoStudents*, *salary*, *annualWorkingDays*), Student/Trainer (*studentsPerTrainer*, *calculateSPT*). The rule that expands the task model is depicted in Figure 7-22, which states: For each task that manipulates a domain class, a new subtask is created for each attribute. After applying the transformation rule each of the new sub-tasks will be mapped on the corresponding attribute of the class. The resulting task model is depicted in Figure 7-23.

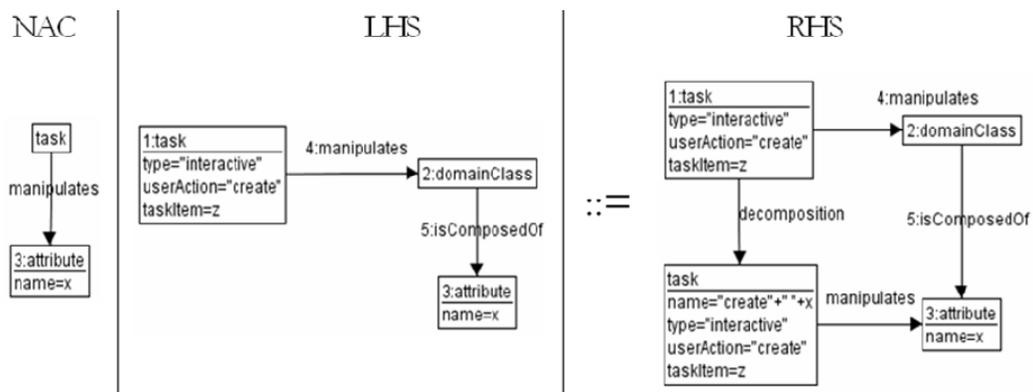


Figure 7-22. Rule for Consolidation of the task Model.

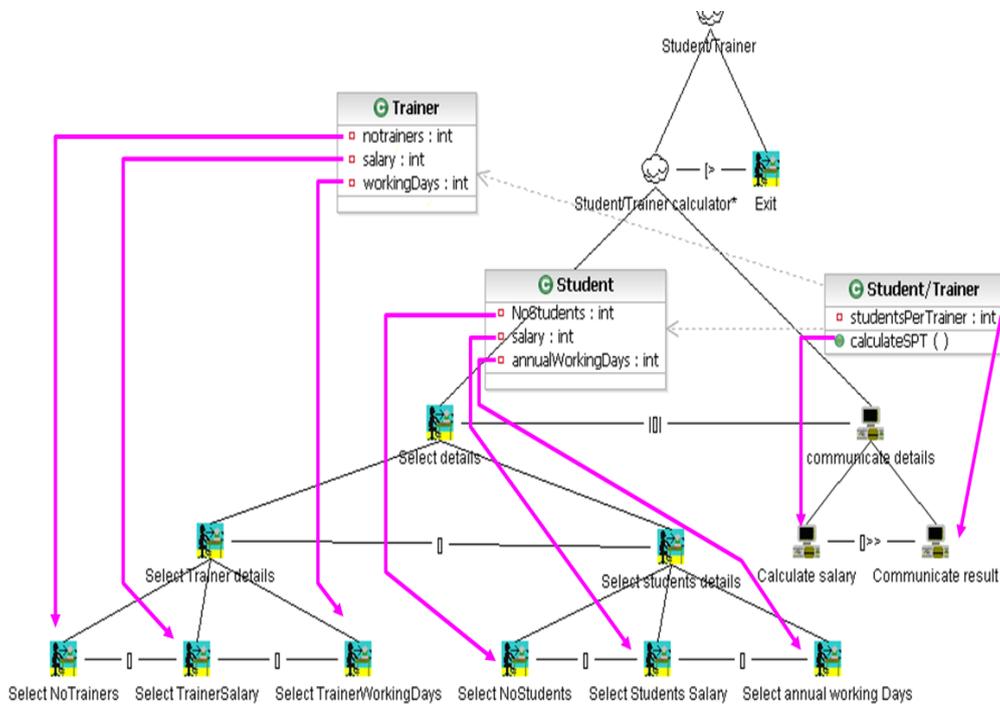


Figure 7-23. Consolidated Task Model of the student/trainer system.

7.1.2.b Step 2: From Task and Domain Models to Abstract UI Model

An Abstract User Interface of the of the student/trainer system in IdealXML [Mont03] in Figure 7-26 the big square denotes an abstract container (AC). For the example all task that are not leafs became ACs. They were obtained using the following transformation rule (Figure 7-24): for each task 1:Task decomposed in any other task 2:Task the Left Hand side (LHS) is the rule is matched then the right hand side (RHS) creates for the task 1:Task a relationship that tells that the task is executed in an AC. Negative Application Conditions (NAC) determine conditions to prevent the rule to apply. In this case the NAC prevent infinite loops because if a task has been already assigned with a relationship IsExecutedIn an AC then no new relationship is created.

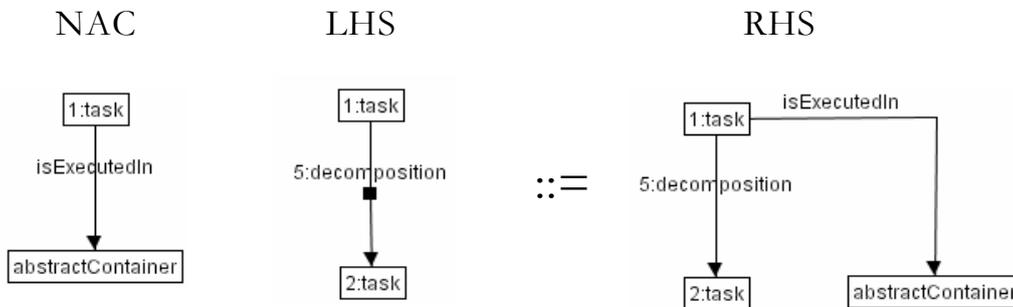


Figure 7-24. Rule for abstract container assignation for tasks.

The abstract individual components (AIC), depicted inside the big rectangles in Figure 7-26, corresponds is an abstraction that allows the description of interaction objects in a way that is independent of the modality in which it will be rendered in the physical world.

An AIC may be composed of multiple facets. Each facet describes a particular function an AIC may endorse in the physical world. Four main facets are identified: an input facet describes the input action supported by an AIC, for instance selecting a value from a range; an output facet describes what data may be presented to the user by an AIC, for instance, showing the summary of the bank transfer to a client; a navigation facet describes the possible container transition a particular AIC may enable, for instance, each navigation arrow in a browser has a navigation facet; and a control facet describes the links between an AIC and system functions i.e., methods from the domain model when existing. For this example task the prevalent task of the system is the selection of a value then the AIC must have an input facet.

The transformation rule to create this facet on each AIC (Figure 7-25) creates this facet in the RHS side when the rule match LHS a task that manipulates an ele-

Chapter 7. Validation

ment where the actionType attribute of the task is select. We will come back later to this attribute of the task that play an important role in further transformations.

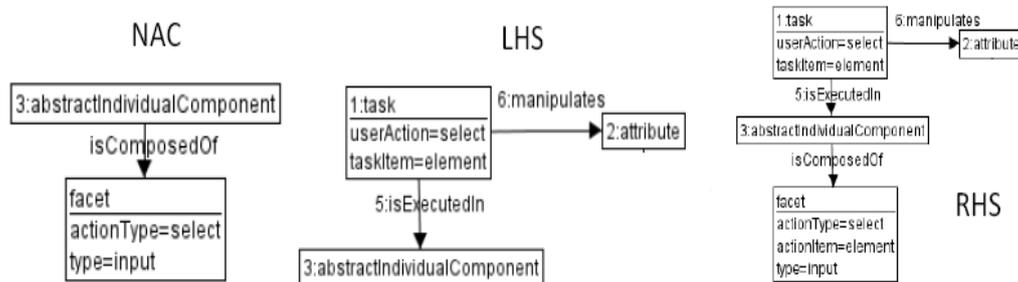


Figure 7-25. Each leaf task is executed in an abstract individual component.

In this example no navigation facets are needed but for the system tasks AICs with control (🔧) and output (🔑) facet were created. The final result of the rules applied to the task model to obtain an AUI is depicted in Figure 7-26.

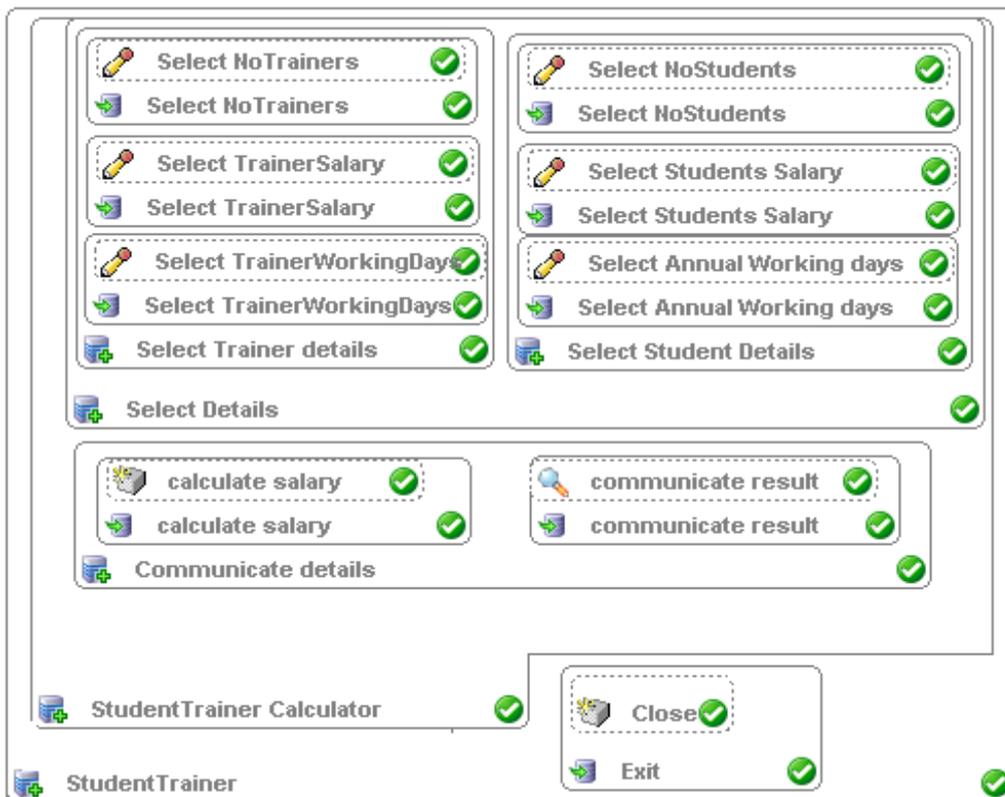


Figure 7-26. Abstract User Interface Model.

7.1.2.c Step 3: From Abstract to Concrete User Interface Model

The next step of the transformation consists on transforming the AUI into CUI. The resulting concretization from the consolidated task and AUI models is depicted in Table 7-2. In the example even that a textfield can be used for the selection value a slider is more suitable due the existence of maximum and minimum values for the domain data.

Abstract Interaction Component	Task Type	Task Item	User category	Possible Concrete Interaction Component
Select NoTrainers	Select	Element	Interactive	A slider, text field
Select Trainer Salary	Select	Element	Interactive	A slider, textfield
Select TrainerWD	Select	Element	Interactive	A slider, textfield
Select NoStudents	Select	Element	Interactive	A slider, textfield
Select Student Salary	Select	Element	Interactive	A slider, textfield
Select annualWD	Select	Element	Interactive	A slider, textfield
Calculate Salary	Start	Element	System	Control
Communicate Result	Convey	Element	System	Output Text component
Exit	Terminate	Collection	Interactive	A trigger

Table 7-2. Concrete Interaction Object selection for the student trainer case study.

The transformation rules used for this step use the attributes of the AUI action type and action item. In our example so far the user action is of type select and the task item is an element (Figure 7-25). Combining these attributes and attributes from the input facet (input card Min and max denoting a range of values) then a slider is selected for each AIC (Figure 7-27). The element to show the result of the calculation is transformed into an output text (Figure 7-28).

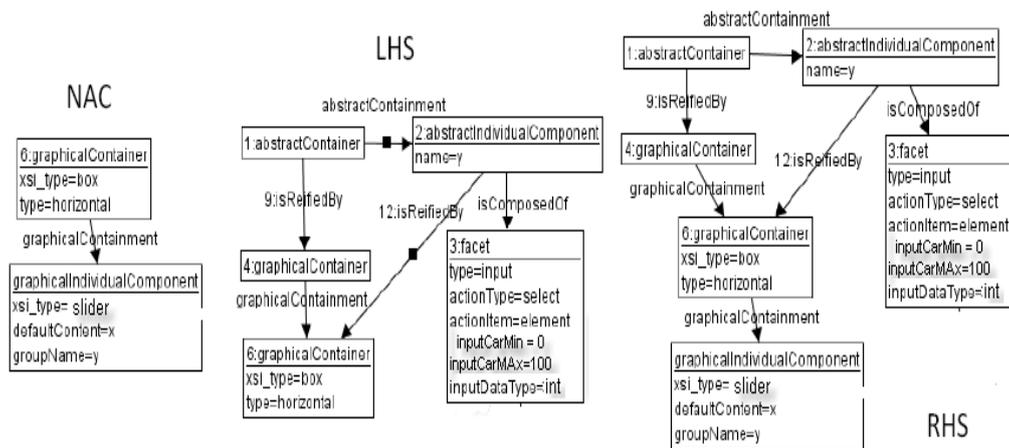


Figure 7-27. Mapping rule for slider selection.

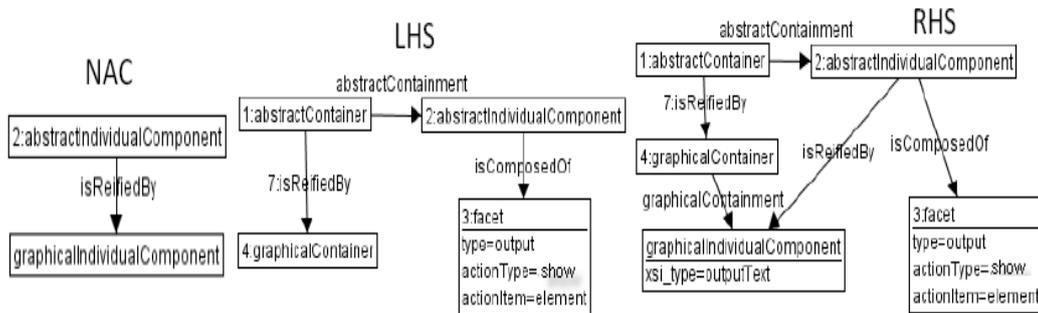


Figure 7-28. Mapping rule for output text selection.

Adding the behaviour to the TDCIO. Using formalisms to model interaction in virtual environments has many benefits: allows measuring the impact of changing input devices and/or interaction techniques before actually implementing them, detecting similarities and dissimilarities in the behaviours, and evaluating the effects of these dissimilarities in the prediction of user performance [Neal01]. In our case we rely on ECA rules to specify behaviour. The event model (Figure 3-20) provides the necessary information about possible user interactions. In this case the example shows the meta-model two event sensors. The sensor is compatible with the definition promoted by the standard Extensible 3D (X3D) for cross-platform, inter-application 3D content delivery. In our example, the event is set to the sliders and any modification of the value affects the result. There is no particular condition to be satisfied in any slider. Since the boundaries are automatically checked (min, max values). The min and max values come from the domain model and their existence determined the selection of a slider as a CIO. Each sensor has its own code to calculate the appropriate value depending on their min, max value. Any change on any slider will then trigger an event to the main sensor that is connected with all the sliders. This event modifies the value of the student/trainers object. The behaviour was set Vivaty Studio, every slider has a sensor connected to it, this sensor evaluates applies when there is a translation and modifies the current value of the slider. By analogy the rest of the sensors are interconnected with the corresponding objects.

7.1.2.d Step 4: From the concrete to the Final User Interface

The last step is to generate the code for the 3D scene. Vivaty studio support several file type options for the final execution of the 3D world. In our case we were just interested in X3D and VRML code. The final result of our method can be seen in Figure 7-29.

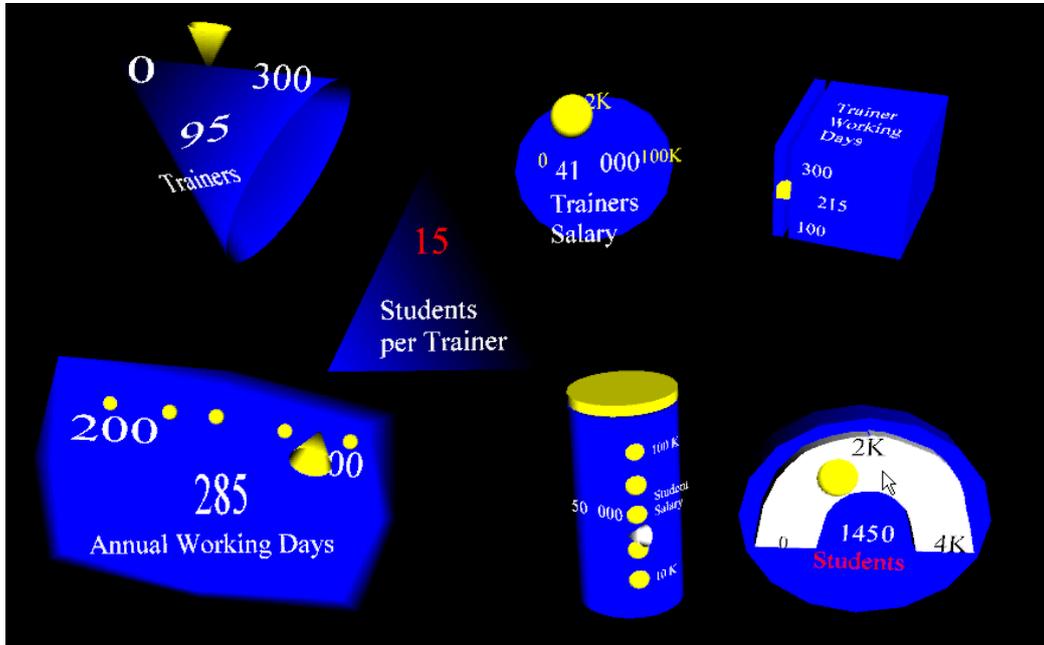


Figure 7-29. A final rendering of the of the student/trainer system.

7.1.3 Case Study 3: Advanced Flight Management System handled by a 3D Navigation Display

In this section the most complex case study used to illustrate our methodology is presented. It refers to the modelling of a 3DUI for to control the navigation display (ND) of the Advanced Flight Management System (AFMS) of an aircraft. This has been done in the context of the European project Human. The methodology used in this dissertation has been proposed and reported [Huma08] for UI modelling.

7.1.3.a Problem Scope

The AFMS is a piece of software that helps pilots to manage their flight in term of trajectory production (e.g. generate trajectories out of a constraint list) [Huma08]. The AFMS is handled via the Advance Human Machine Interface (AHMI) by pilots. The AHMI consist of a Navigation Display (ND), the Control Display Unit (CDU), and the Central Human Interface and Management Environment (CHIME) [Huma08]. The interaction between the pilot and the AHMI is through the different User Interfaces (UIs) that composed the ND and the CDU. Due to its complexity and criticality the AHMI development requires an interdisciplinary approach and a profound theoretical background, traditionally covered by a method. An AHMI development method is expected to facilitate the design of

usable AHMI systems. How we applied our method to the design of the ND has been described in [Huma08]. The objective is to describe the design of the ND for the AHMI to be used by the Virtual Simulation Platform (VSP) [Huma09].

7.1.3.b Software tool chain

The SAHMI is related to a cognitive architecture (CA) that is in charge of simulating pilot's behaviour. There is a constant communication between these two pieces of software as the CA interacts with the SAHMI and not with the real system. The tool chain, including the CA, is depicted in Figure 7-30. The task of the SAHMI is to monitor and send messages, related to AHMI interaction, to the CA. This means that to execute an action over the AHMI the CA sends a message to the data pool indicating the triggered event (e.g. click on negotiation button). Then, the SAHMI reads the message from the Datapool and analyze its content in order to provide a feedback to the CA. Based on the action to perform, the status of the CHIME and the status of the Datapool the set of action is triggered internally (e.g. change state of the system to indicate negotiation) to update the CHIME and Datapool. Finally the feedback, a set of messages with information related to the visual feedback resulting from this action, is sent to the Datapool. The messages are accessed by the CA.

7.1.3.c Software architecture

In Figure 7-31, the SHAMI architecture is shown. A repository with UsiXML formalism describing the AHMI is used. These specifications can be edited using any text editor; a XML-editor is preferable but not mandatory. This file is read using a parser that validates the specification and transforms this into a machine readable structure called model merger. The UI is complemented with data accessed from the data pool or the CHIME. From the Datapool messages from CA are retrieved while the CHIME client collects information about the status of the system. Both sources of information are processed in the model merger and a message for the CA is sent to the Datapool client. Finally the data from the data pool and the CHIME must be transformed to be compatible with UsiXML format in order to store a log File history of UI evolution to be used for UI evaluation.

7.1.3.c.1 XML Parser

The UsiXML is a xml-based language and order to use this formalism a XML Parser is needed. The parser reads a UsiXML file a load it into memory. Then, the file is converted in an XML DOM object that can be accessed and manipulated. Most libraries contain functions to traverse XML trees, access, insert, and delete nodes (elements) and their attributes.

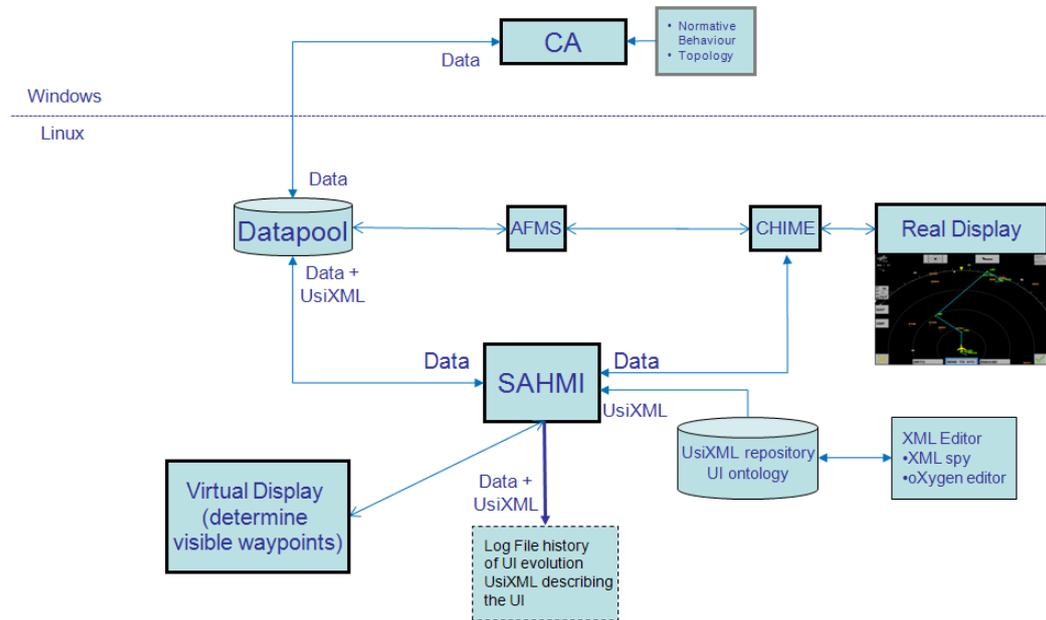


Figure 7-30. Tool Chain of the Symbolic AHMI.

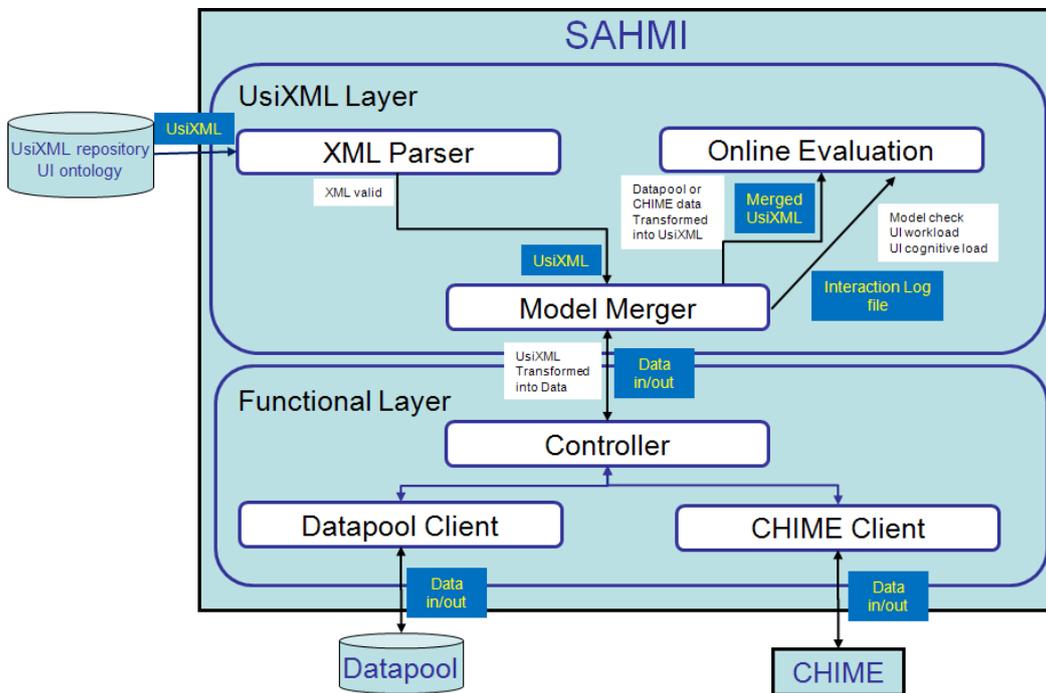


Figure 7-31. Symbolic AHMI Architecture.

7.1.3.c.2 Online Evaluation

The evaluation layer of the SAHMI keeps a trace of the evolution of the UI during the interaction. The Model Merger layer reconstructs the UsiXML and sends it to store it in the online evaluation tool. Such evaluation can be automatically evaluated with the Usability Adviser [Vand06] that is a tool to determine the ergonomics characteristics of a UI written in UsiXML. This tool evaluates ergonomic rules to determine workload, visual obstruction.

The evolution of the evaluation tool considers the possibility to add another module, transformer in Figure 7-32, that will modify the UsiXML file propose some possible changes to the UI and send this modified file to the evaluation tool. Over time it might be possible to have two files (the original and modified) and compare their resulting evaluation.

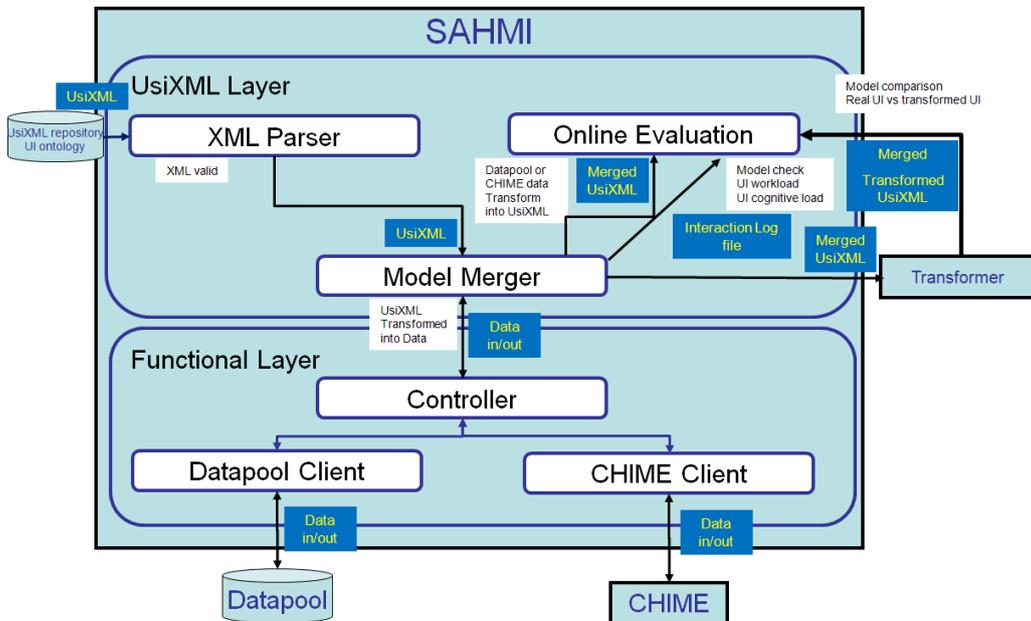


Figure 7-32. Evolution of the Symbolic AHMI Architecture.

7.1.3.c.3 Model Merger

The Model Merger module is responsible to integrate the data coming from the controller, data from the two clients: DP and CHIME, and adequate a UsiXML correspondence to such data and send it to the evaluation tool. This module is also in charge of transforming the UsiXML file into a data format that is compatible with the controller module.

7.1.3.c.4 Controller

This module is in charge to synchronize information coming from the DP client and the CHIME client; it sends data to them as well. This module just deals with the DP and CHIME format.

7.1.3.d Step 1: Task Modelling

There are more than 50 direct actions that can be manipulated on the AHMI. As there is no significant difference on what it corresponds to UI objects and layout. We will restrict this case study to one task, although, the rest of the UI can be generated by analogy. The task that we will focus is the *generation of a new trajectory* in the air (Figure 7-33). To generate a trajectory in the air, the user has to select a waypoint (wpt) on the constraint list to which the aircraft shall fly directly and at which it shall intercept the constraint list. The AHMI automatically suggests a *suitable wpt* that is written in a field above the DIRTO button, whenever the mouse pointer is moved over that button. By pressing on the field above the DIRTO button, the user accepts the suggestion (*trigger suitable wpt*). After clicking on the waypoint or the field with the suggested waypoint's name, a trajectory leading from the current position to the intercept point and from there on along the constraint list is generated (system tasks of the subtree *create arbitrary trajectory*). While the constraint list is shown as a blue line, the trajectory is shown now as a green dotted line.

To select another waypoint, the user simply has to click first on the DIRTO button (*create wpt*) and then move the mouse onto the waypoint on the constraint list he wishes to select. The waypoint's name is then marked in yellow and written on the DIRTO button (*select arbitrary wpt*). Special attention must be taken to the calculate trajectory feedback as more than once a WP can be selected then if one WP was selected a trajectory is proposed but if another WP is selected then the previous trajectory is deleted and the new proposed trajectory is drawn.

After the trajectory has been generated, it can be negotiated with ATC simply by moving the mouse over the SEND TO ATC menu. A priority could be chosen during the negotiation process with ATC (*select negotiation type*). After selecting the negotiation type the system *show* the *feedback* from ATC about the trajectory.

Thereafter, even if the negotiation has failed, a click on ENGAGE! (*trigger trajectory engage*) activates the AFMS guidance, which generates aircraft control commands to guide the aircraft along the generated trajectory. The trajectory is then displayed as a solid green line (*show trajectory*). If the trajectory is approved by ATC and engaged, i.e. the AFMS guides the aircraft along that trajectory, the dark grey background of the trajectory changes to a bright grey one.

One relevant aspect of relying on task models revealed a usability problem on the existing system. The current version of the AHMI allows pilots to trigger any of the three actions (select, negotiate and engage trajectory) without forcing a logical sequence of the tasks. Interaction objects are enabled even that they should not be. The task model structure and task model relationships assures, at some point, to consider the logical sequence of actions as constraints for the further concretization of the tasks.

7.1.3.e Step 2: Abstract User Interface Modelling

The identification of the AUI structure is ensured by applying Rule 7-3, Rule 7-4, Rule 7-5, Rule 7-6, and Rule 7-7. The selection of the AIC is based on rule 7-8. Spatio-temporal arrangement of the AIO is done with rules 7-9, 7-10, 7-11, and 7-12. Facets are added with rules 7-13. The transformation from task to AUI model is depicted in Figure 7-34. No particular difference exists from the rules used to derive the AUI compared with previous case studies. Only one interesting advantage of defining the AHMI as an AUI model is the fact even that the AHMI is the digital version to interact with the AFMS, it is possible to do it via a physical interface. Modelling the UI in terms of an abstract container allows the design of modality independent interfaces.

7.1.3.f Step 3: Concrete User Interface Modelling

The concretization of the AUI into a concrete user interface model requires more than just relying on existing transformation. Due to the fact that the AHMI core functionality does not correspond to a traditional UI, an extension of the CUI model were need to include those new concepts. The list of elements identified so far (Figure 7-35) are:

- *map*: is a CIO dedicated to display the position of the aircraft, airport, trajectory, nav aids, etc. It has several attributes such as: *viewMode* (plan and monitor mode), flags to determine whether *isWaypointOnEnrouteAirwaysVisible*, *isNavAidsVisible*, *isenrouteAirwayVisible*, *isSIDVisible*, *isSTARVisible*, *isApproachVisible*, *isConstraintListVisible*, *isActiveTrajectoryShow*, *isTubeVisible* and *isRestrictedAreaVisible*.
- *verticalProfile*: is the vertical view on the ND. It has attributes such as: *distanceScale* (for instance, nautical miles), *distanceSteps* to define steps between the beginning and the end of the distance, *timeScale* (the format of the time), *timeSteps* to define steps between the beginning and the end of the distance, *CFLPosition* (Cruise Flight Level), *intPosition* (INT-line the intercept altitude at which the localizer is intercepted). The vertical profile is composed by two other objects:
 - *speedProfile*: represents the speed for the trajectory of the aircraft.
 - *altitudProfile*: is a line with a specific *colour* corresponding to the altitude *scale* at the left edge, which shows hundreds of feet.

Chapter 7. Validation

- *lateralProfile* is the bird view of the aircraft, it is composed of a *compassRose* on which the *currenttrack*, refers to the id of the current track that the A/C is flying, is displayed; the *distanceLines* denotes the distance between the circles of the compassRose.
- *objectOnMap*, this object has a *position* and is specialized in: *trajectory*, *constraintList*, *waypoint*, *onEnrouteAirway*, *aircraft*, *navaid*, *airport*, *enrouteAirway*, *SID*, *STAR*, *approach* and *restrictedArea*

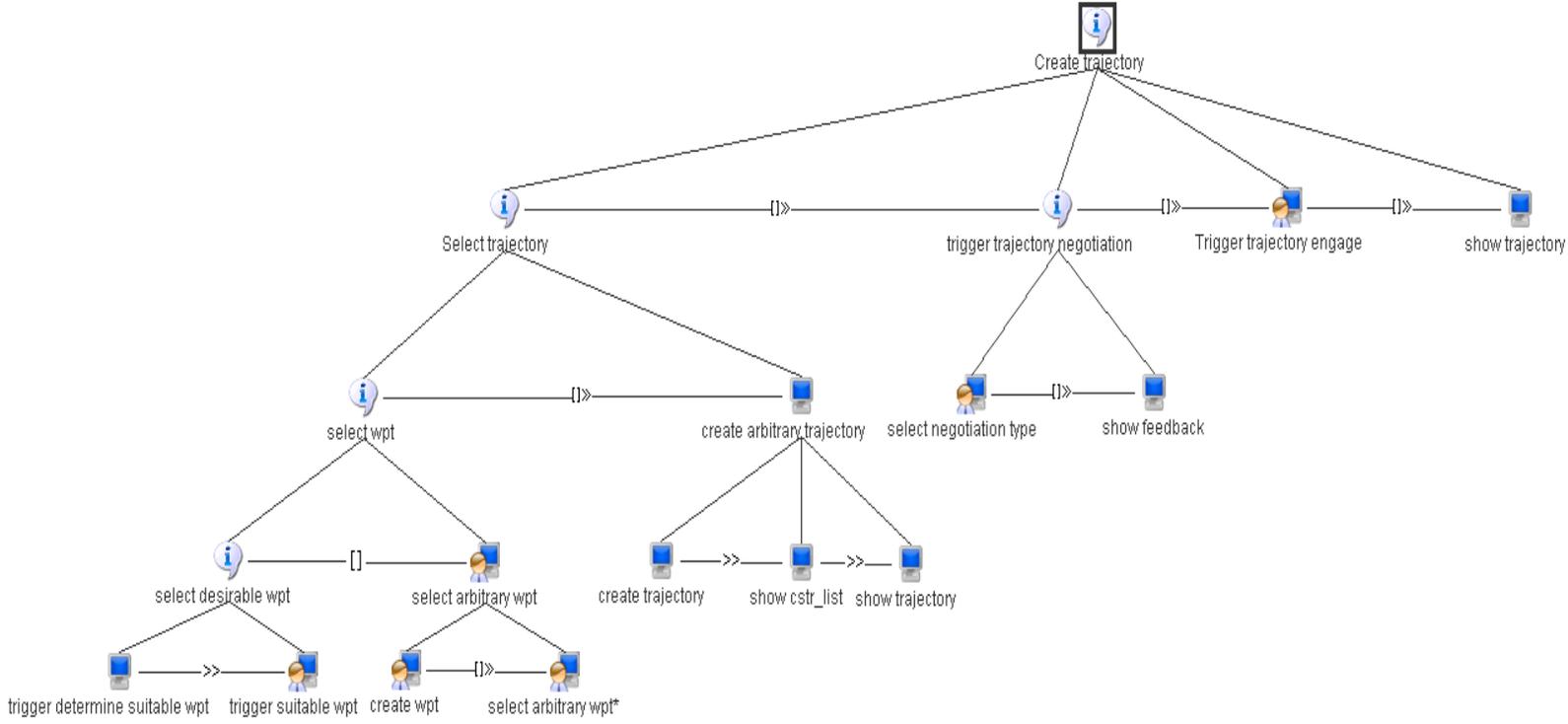


Figure 7-33. Task Model of the create trajectory task.

Chapter 7. Validation

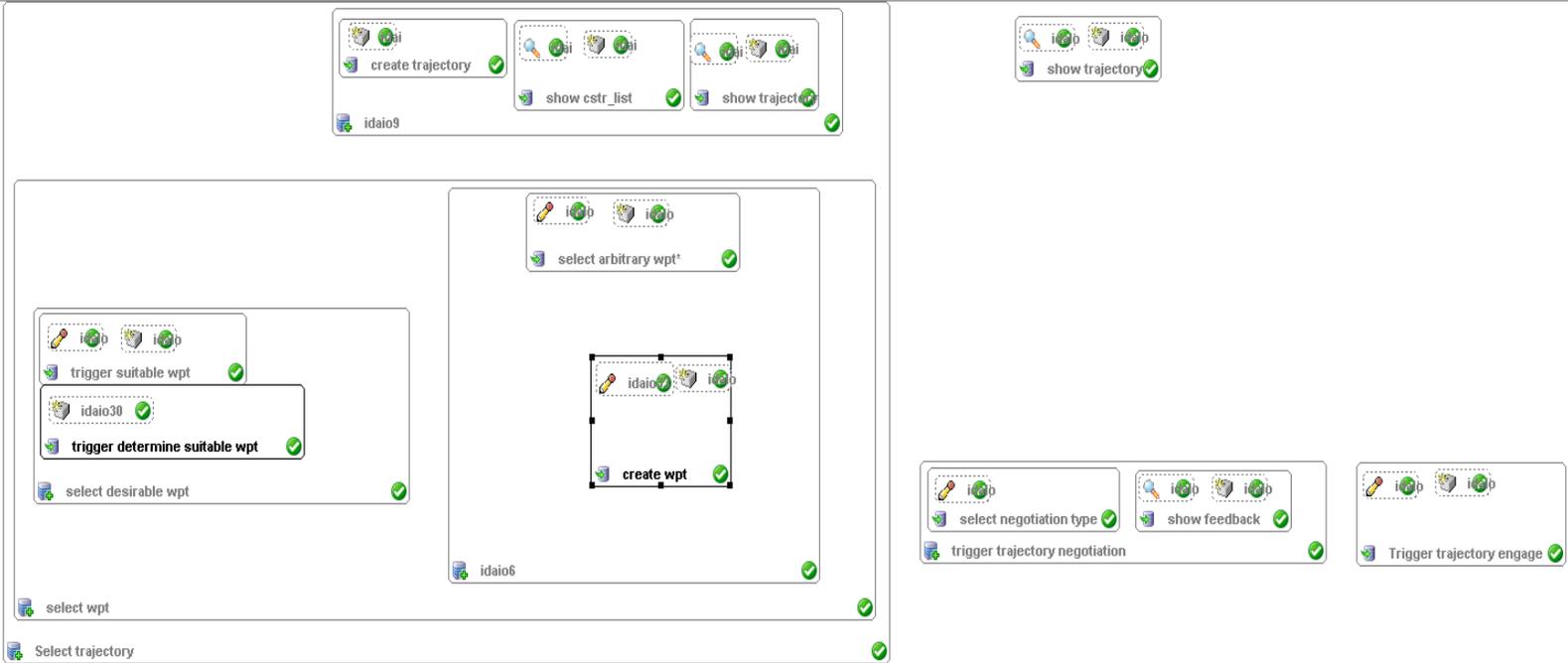


Figure 7-34. Abstract User Interface of the select trajectory task.

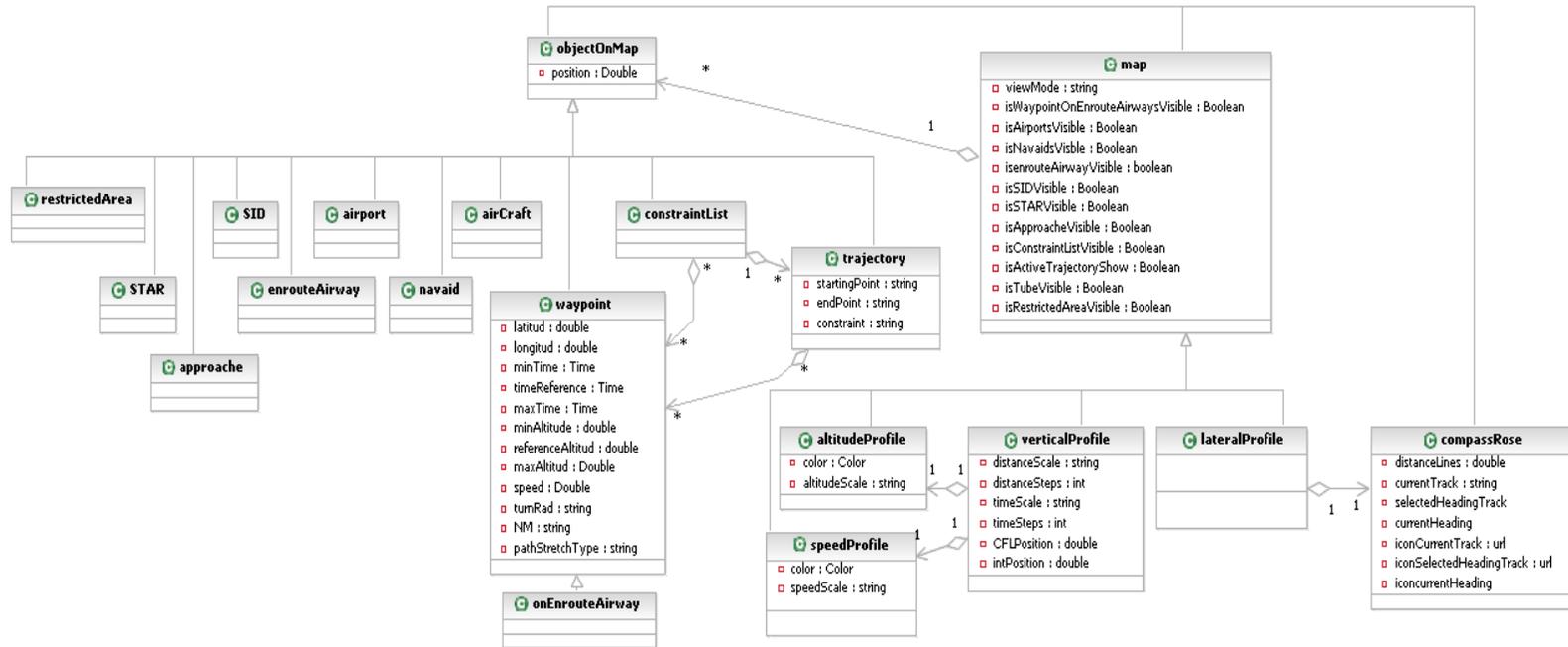


Figure 7-35. Special Graphical Individual Components used in the AHMI.

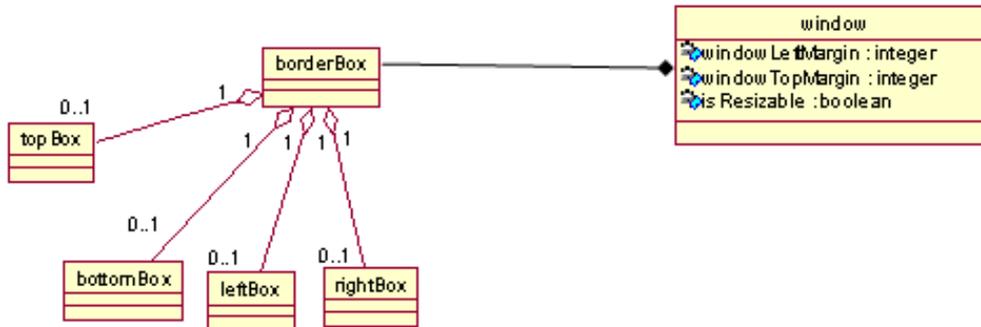


Figure 7-37. Semantics of the CUI model of the Navigation Display layout.

The corresponding syntax of the layout is as follows:

```

<window input id="w1" name="Window1"
  <borderBox id="bB" name="borderBox1">
    <topBox id="tB1" name="topBox1"/>
    <bottomBox id="bB1" name="bottomBox1"/>
    <leftBox id="lB1" name="leftBox1"/>
    <rightBox id="rB1" name="RightBox1"/>
  </borderBox>
</window>
  
```

At the upper edge are the buttons that control the view mode, e.g. lateral or vertical view.

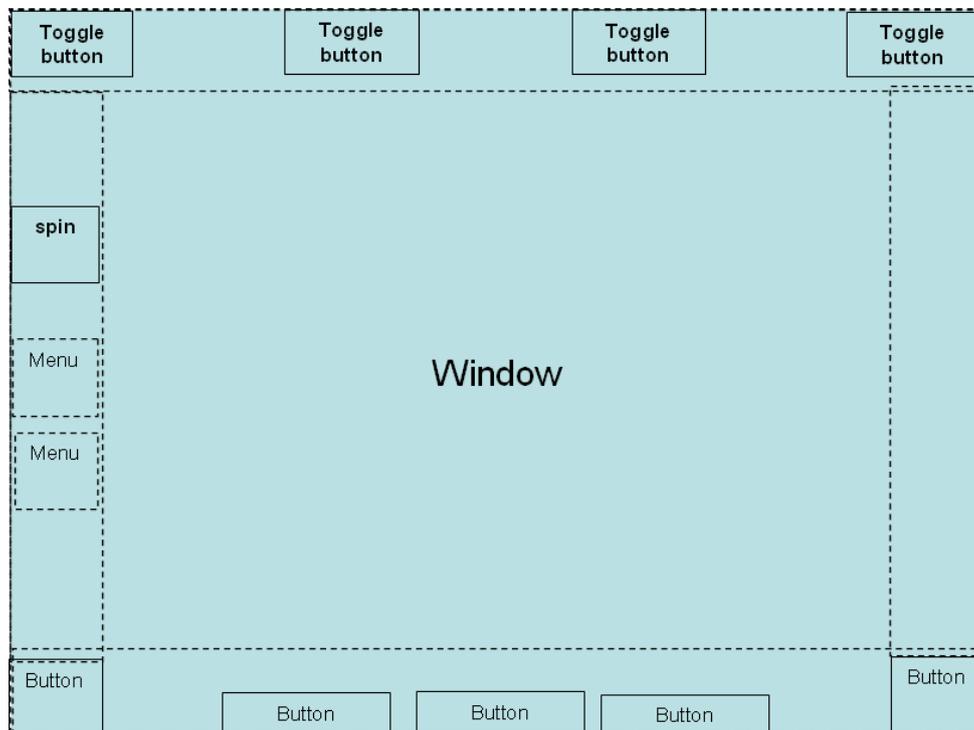


Figure 7-38. Mock-up of the Navigation Display layout.

Chapter 7. Validation

At the left edge are the buttons that control the display mode, e.g. the range of the map or what kind of information is shown on the map. The buttons that control the generation and negotiation of a trajectory are at the lower edge of the display. The buttons at the right edge and in the upper right corner of the display are used for creating and editing constraint lists. Additionally, there is a “cancel”-button in the lower left corner of the display with a yellow cross on it and an “accept”-button in the lower right corner with a green check mark on it. Error messages are displayed in red-framed boxes in the upper left part of the display and are removed by clicking on them or on the accept button. The abstract view of buttons distributions is depicted in Figure 7-38. In excerpt of the ND that has been exemplified so far, the mappings from AUI to CUI model are assured by the application of transformation rules, analogous to the one depicted in Figure 7-27 and Figure 7-28, with the difference in the task type and the facets associated to the controls.

Abstract Interaction Component	Facet Specification	Information to take into account	Possible Concrete Interaction Component
“trigger determine suitable wpt”	Control	Attribute	Label on a button
“trigger suitable wpt”	Control, input	Feedback	button
“create wpt”	Control, input	Feedback	button
“select arbitrary wpt”	Control, input	List of waypoints associated to this task	Waypoints on ND
“Create trajectory”	Control	Attribute	None
“show cstr-list”	Control, output	Feedback, action associated to the constraint list.	Constraint list on ND
“show trajectory”	Control, output	Feedback, action associated to the trajectory	Trajectory on ND
“select negotiation type”	input	Enumerated list of selection types associated to this task	Menu
“show feedback”	Control, output		Dialog Box
“trigger trajectory engage”	Control, input	Feedback	Button

Table 7-3. Mapping from AUI to CUI models.

Behaviour specification. The canonical action types have been shown practical to be used for 3DUI generation. In this particular problem they were used to name the tasks of the SAHMI, as detailed in the guidelines for task modelling (Section 4.2.2). So AHMI user interface actions were analyzed in detail to determine relevant interactions, from the cognitive architecture point of view, with the AHMI. Due to the large number of actions an example is used to illustrate this process.

Chapter 7. Validation

To perform the task *select + priorities* during the downlink of trajectory indicates that after the trajectory has been generated, it can be negotiated with the air traffic controller (ATC) simply by selecting the type trajectory on the SEND TO ATC menu. A priority could be chosen during the negotiation process with ATC. There are five priorities: Normal, Emergency, Technical, Weather, and Scheduling/Traffic.

For this particular task the behaviour formalisation refers to the way to express the functional part of the requirement. As described in section 3.5.4, the *behaviour* is the description of an event-response mechanism that results in a system state change. The specification of behaviour may be decomposed into three types of elements: an *event*, a *condition*, and an *action* (ECA rules). The ECA rules are expressed as algorithms. The *condition* (including system states) are expressions in the format if then else. The *action* is method calls in the body of the algorithm. The event is always a mouse click.

For this example just the schedule behaviour modelling is presented, the others can be generated by analogy, as they keep the same structure. Two variables are involved, which are:

Name	0	1	2
negotiating_trajectory	Mouse_Off	Mouse_Over	
Trajectory	Unselected	Selected	Negotiated

Algorithm negotiate_schedule_trajectory

Input: **ahmi_system_variables_struct, wpt, constraint_List**

Output: **ahmi_system_variables_struct updated**

```
AS ← ahmi_system_variables_struct
AS → negotiating_trajectory ← Mouse_Over
if (AS → negotiating_trajectory is Mouse_Over) then
  if (AS → trajectory is Selected) then
    FeedbackFromATC ← negotiate_trajectory_with_ATC (Constraint_List, wpt, "schedule")
    change_Object_BackgroundColor (sendToAtc_Schedule_menuItem, blue)
    change_Object_BackgroundColor (sendToAtc_Schedule_menuItem, gray)
    AS → trajectory ← Negotiated
    showMessage (FeedbackFromATC)
  else
    showMessage ("No Trajectory is Available")
pop_down_menu (negotiate_atc_menu)
```

This algorithm is mapped to our behaviour model as follows:

```
<behavior id="ID01">
  <event id="ID01" eventType="click" eventContext="menuItem_negotiating_schedule_trajectory"
  device="mouse"/>
  <condition />
  <action description="String" id="ID01" name=" "/>
    <methodCall methodId="negotiate_schedule_trajectory">
      <methodCallParam componentIdRef="ahmi_system_variables_struct, wpt, constraint_List" componentIdRef="" componentIdRef="$FeedbackFromATC"/>
    </methodCall>
  </action>
</behavior>
```

A *method call* occurs when an *action* is triggered by the *event click* with a *mouse* on the *menuItem_negotiating_schedule_trajectory*. There is no particular *condition* to be evaluated. More than 50 action types have been modelled using this method.

7.1.3.g Step 4: Final User Interface Concretization

The FUI as it is currently used in the real system is shown in Figure 7-39. So far, the example has illustrated the different steps but no constraint has been discussed related to the concretization of the model. Evidently for this particular case study, our first step was to abstract the real system functionality and representation (2D).

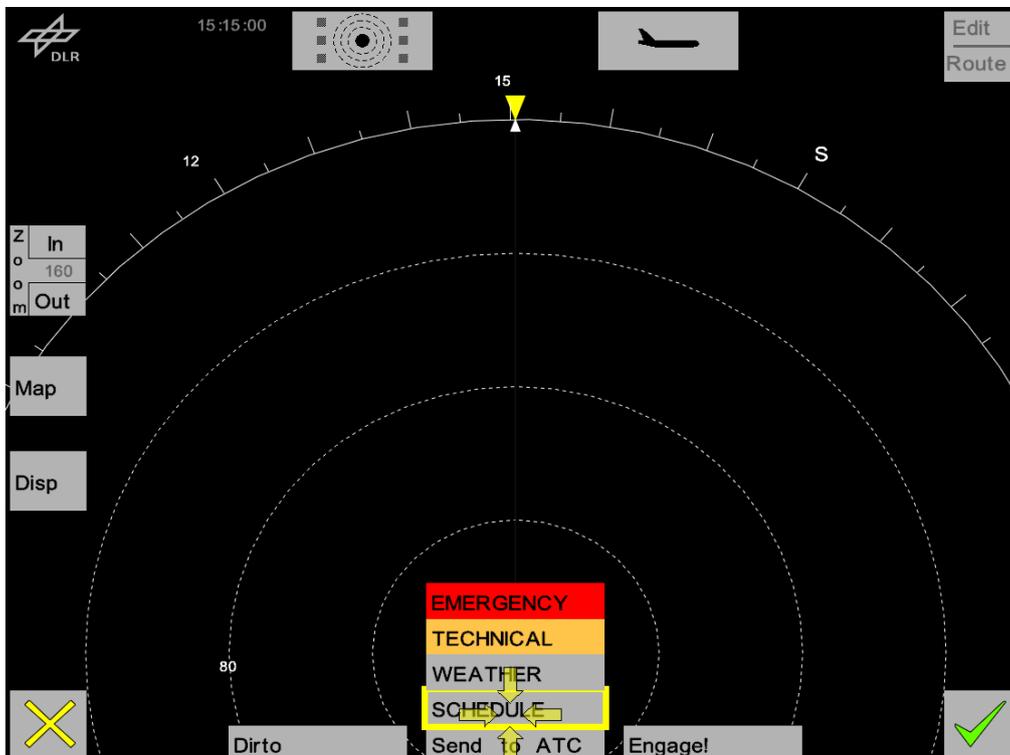


Figure 7-39. AHMI negotiating evolution.

Chapter 7. Validation

An excerpt of the UsiXML code corresponding to the screen shown in Figure 7-39 is reproduced below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by ucl (loc) -->
<window id="w1" name="wLatViewSAHMI" isVisible="true">
  <borderBox id="bB1" name="BBLatView" isVisible="true">
    <topBox ...>
      <bottomBox id="btB1" name="btBLatView" isVisible="true" >
        <Button id="Bu1" name="BuReject" icon="reject.png" isVisible="true" xpos="" ypos=""/>
        <Button id="Bu3" name="BuDirto" defaultContent="Dirto" isVisible="true" xpos="" ypos=""/>
        <menu id="Bu2" name="BuNegotiation" defaultContent="Negotiation" xpos="" ypos="">
          <menuitem id="Bu2" name="mISendToATC" defaultContent="Dir To" xpos="" ypos=""/>
          <menuitem id="Bu2" name="mISchedule" defaultContent="Dir To" xpos="" ypos=""/>
          <menuitem id="Bu2" name="mIWeather" defaultContent="Dir To" xpos="" ypos=""/>
          <menuitem id="Bu2" name="mITechnical" defaultContent="Dir To" xpos="" ypos=""/>
          <menuitem id="Bu2" name="mIEmergency" defaultContent="Emergency" xpos="" ypos=""/>
        </menu>
        <Button id="Bu3" name="BuSendToATC" defaultContent="Send To ATC" isVisible="true" xpos="" ypos=""/>
        <Button id="Bu4" name="BuEngage" defaultContent="Engage" isVisible="true" xpos="" ypos=""/>
        <Button id="Bu5" name="Buaccept" icon="accept.png" isVisible="true" xpos="" ypos=""/>
      </bottomBox>
    <leftBox ...>
    <rightBox .../>
  </borderBox>
</window>
```



Figure 7-40. Navigation Display rendered as a 3DUI.

Chapter 7. Validation

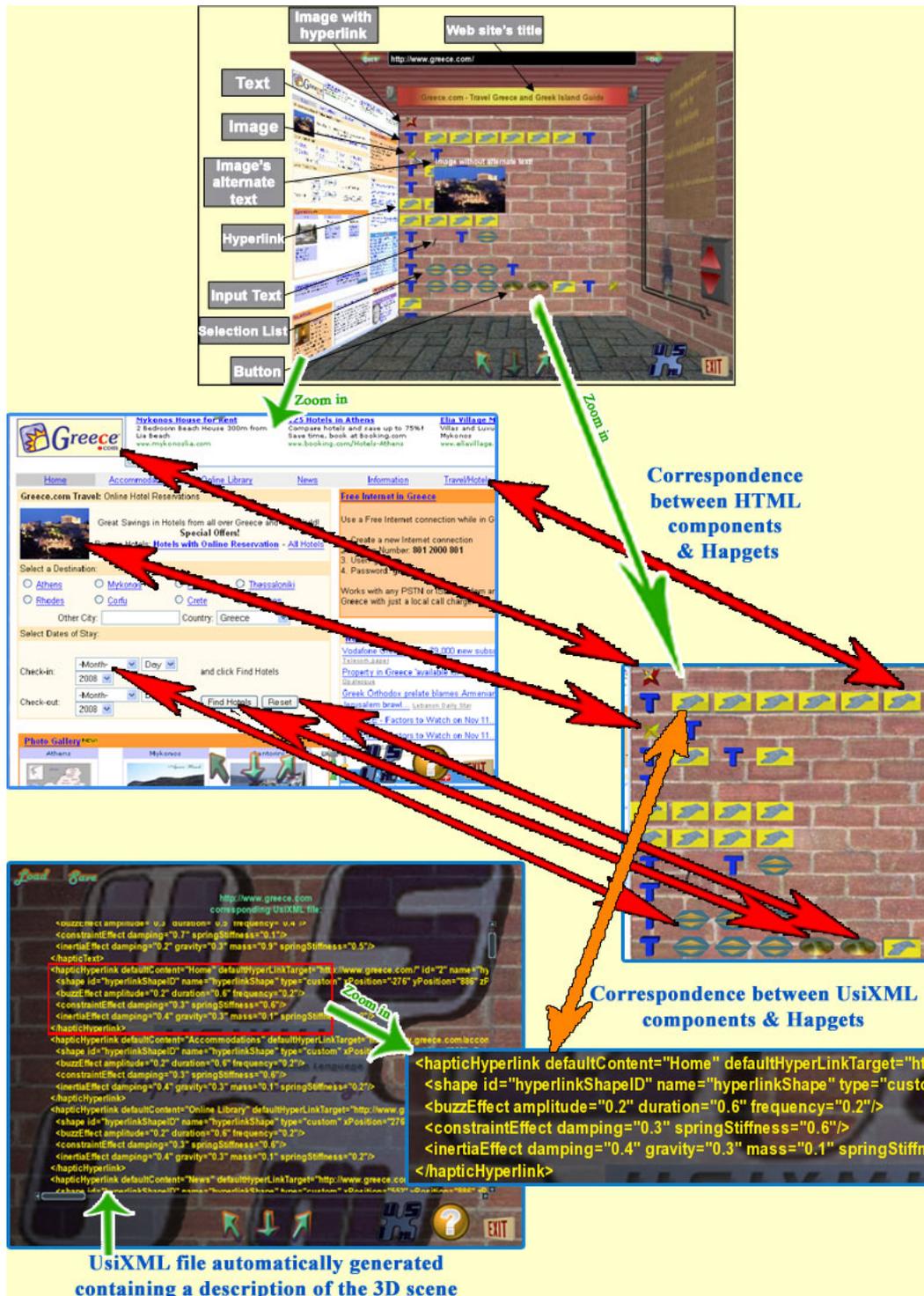


Figure 7-41. Test case: www.greece.com [Kak108].

Chapter 7. Validation

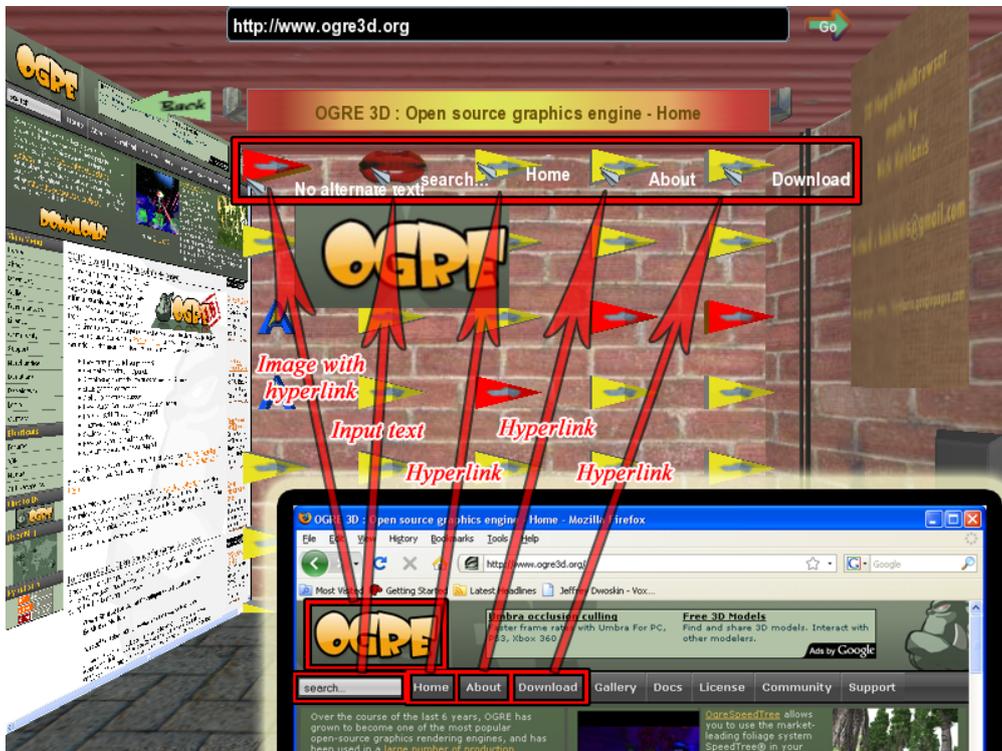


Figure 7-42. Case study: www.ogre3d.org.

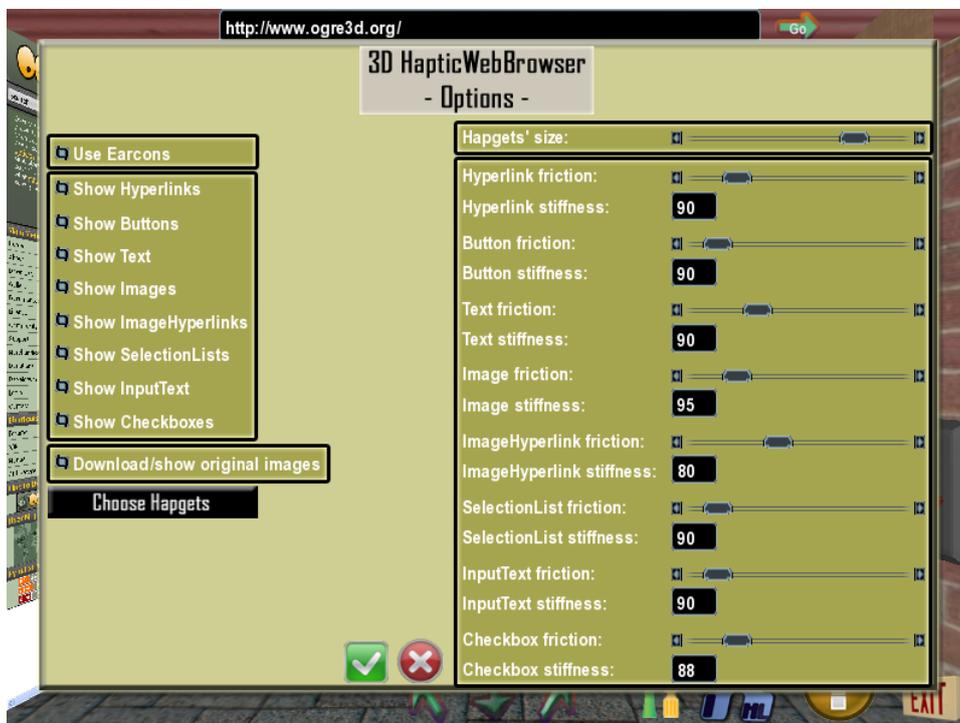


Figure 7-43. Options.

Chapter 7. Validation

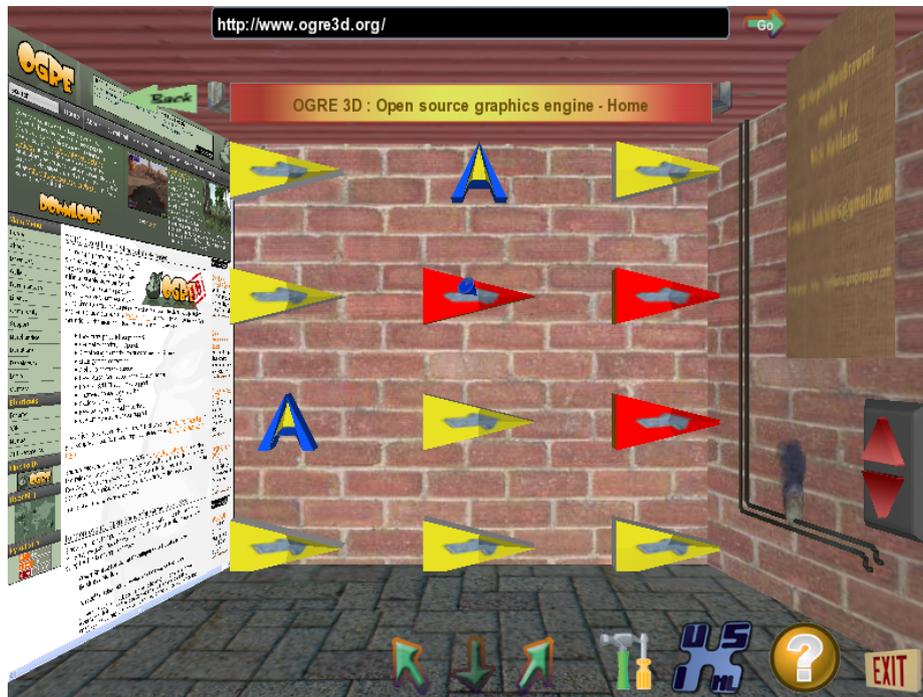


Figure 7-44. Large Hapgets.

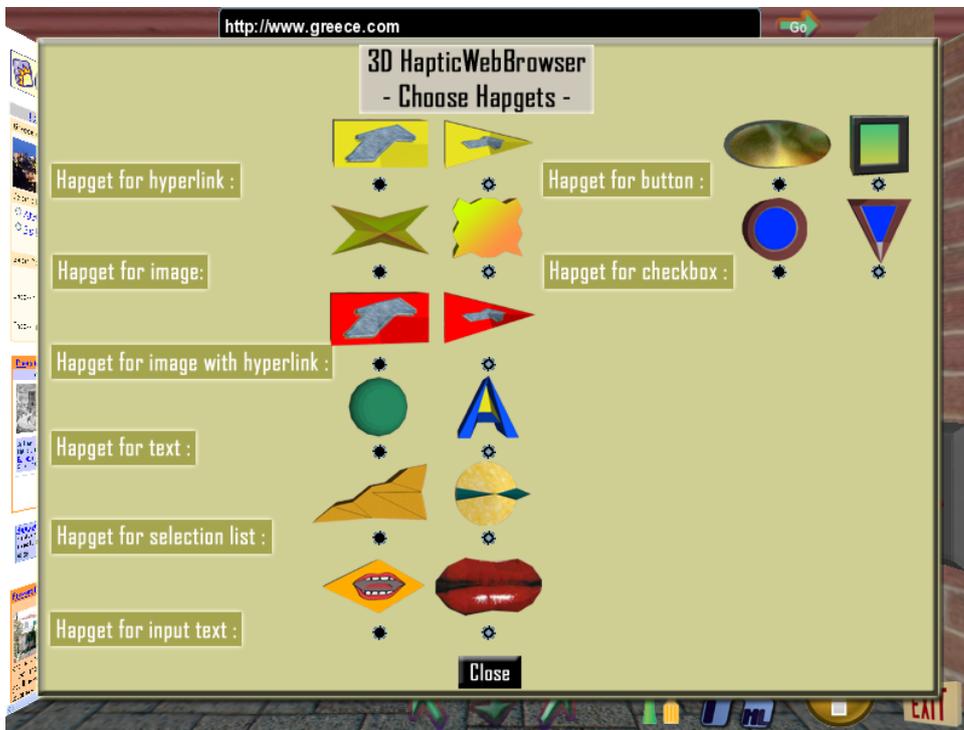


Figure 7-45. Choose hapgets.

The final rendering of the AHMI in 3D (Figure 7-40) right now is just about the presentation. Although, it is out of the scope of this work further investigation will be conducted to evaluate the impact of this representation. Even more, the navigation compass rose would have an impact or not?

7.1.4 Case Study 4: Haptic application

A significant case study is presented aimed at offering simultaneously visual and haptic interaction. The advantages of such combination offer some advantages. On the one hand the visual channel is preferred by users experiencing no problem in using it; on the other hand, the haptic channel could be used by impaired users, and the combination of both modalities could be offered to those who suffer from sight impairment, but who are not blind: when the visual impairment is important, the tendency would be to rely more on the haptic interaction (haptic dominant interaction) as opposed to the visual interaction when the visual impairment is light, but existing, thus, filling one goal of this dissertation.

This case study presents a forward engineering from CUI to FUI mapping. The models have been already presented and discussed in section 3.5.7. In the 3D scene every HTML component has a 3D representation, a description and an earcons [Brew98; Myna94]. For images, with or without hyperlink, there is also a 2D representation, which contains the original image. When the Phantom “touches” an object, the user immediately hears the earcon that corresponds to the objects of this type. If user presses the CTRL button of the keyboard while Phantom is in contact with an object, the object’s description is being heard via the speech synthesis engine. Figure 7-41 presents how the haptic rendering engine works. User starts the speech recognition engine (by pressing the SPACE button of the keyboard) and then gives a URL (“www.greece.com” in this test case) using the microphone or types the new URL using the keyboard (when a key is being pushed, its value is being heard via the speech synthesis mechanism). The corresponding to this URL 3D scene is being created immediately.

At the left of the scene, user can see the web page as it would be presented in a normal web browser. This side of the scene also interacts as a common web browser. For instance, user can click on a hyperlink and go to another URL with simultaneous update of the 3D objects presented in the scene. There are some buttons that give user the opportunity to move in the 3D scene and focus on whatever he/she wants into the scene and many visual effects that make navigation through the internet much more impressive than it is via the typical web browsers.

For instance, when the cursor goes over a 3D image, the original image shows up and the alternate text of the image is following the cursor as a 3D component's tooltip. However, all these features have to do with the users that have normal vision. The functionality that concerns the visual-impaired users is limited to the haptic and the auditory channel. A blind user can only interact with the 3D components via the haptic device (Phantom Desktop), hear all the necessary information via a speech synthesis engine and pass to the application all the necessary input via a speech recognition engine (using a microphone). Another case study is depicted on Figure 7-42 .

There is a set of options that the user can set according to specific needs/preferences (Figure 7-43). User can enable/disable the earcons, exclude some component types from 3D/haptic rendering, set the value of friction and stiffness for each hapget as well as change the size of the hapgets. Additionally, to achieve faster loading, original images' downloading can be disabled, as this feature concerns only the non-blind users.

For a novice user it is recommended to use large hapgets, because as the size of the hapgets become larger, the identification of each hapget using the Phantom becomes easier/faster. The negative point of having large hapgets is that the number of the hapgets contained in the 3D scene at a time becomes smaller and its consequence is the slower navigation in the web page. Figure 7-44 shows a test case where large hapgets have been chosen. User can also choose the hapget to be used for each component type (Figure 7-45).

7.2 Internal Validation

7.2.1 Why should the User Interface development method be formal?

In order to keep consistency between the different tools, to have a common ground for every aspect in the project the method used must be formal at least for the following reasons:

- *Modifiability*: If there is a change in a model then the 3DUI changes accordingly.
- *Complexity*: As the 3DUI is part of a command and control system, it may represent a huge quantity of code. User interface design and construction tools must provide ways to address this complexity as well as the reliability.
- *Safety Criticality*: The 3DUI might be part of a safety critical system, as it is the case of the AHMI modelled in case study 3. In order to warranty and

investigate its behaviour (functionalities), models are needed. The use of a formal specification technique is extremely valuable, because it provides non-ambiguous, complete and concise ways of describing the systems [Barb06].

- *Rigorously*: The development life cycle of 3DUIs must involve the same level of rigor that is typically used in software engineering (SE) [Boda94].
- *Reasoning*: Because from the models describing the 3DUI some reasoning is possible, such as:
 - *Automatic*. Computer based system might analyze data related to the 3DUI automatically and might be able to predict users behaviour.
 - *Production of errors*.
 - *Processable*. Models can be processed and studied by devoted systems.
 - *Checking properties*. Analysis of the different effects produced in the 3DUI by modifying properties of the components, for instance, changing background colour, fonts of labels, etc.
 - *Human readable*. This is not necessarily always achieved but model are expected to be understandable for humans.

7.2.2 Why should the User Interface development method be standardized?

So far, the need of a formal method has been described. The different tools and methods used in the context of 3DUI stress the need of a standard to enable:

- *Communication*. Different tools require a standard communication channel.
- *Consistency*. Different tools require a standard for consistency in the information they exchange. Transferring knowledge, building interfaces between agents (humans or artefacts) is a crucial task for future applications in aeronautics. Focus will be on the exchange of knowledge across application and document format boundaries; a common pool of knowledge is needed where everybody may share and retrieve knowledge [Reis06].
- *Common knowledge*. By relying on a standard different tools might share knowledge about the 3DUI, for instance, task models formulation might include knowledge about the 3DUI, so as, cognitive models might use the same knowledge to predict users behaviour.
- *Scalability* (models can grow over time without affecting significantly the resulting process complexity). Scalability of the approaches to deal with real-life and real size applications are often difficult to prove, due to the size and the number of models that have to be specified and managed [Barb06]. A well-structured model can be increase in a more flexible way.

There have been some attempts to standardize formal methods of some aspect of the 3DUI, notably the CUI model has recommendations from web 3D consortium. The W3C model-based user interface incubator group will include as part as

their recommendation a 3DUI model. We have actively participating in this W3C effort [W3C09].

7.2.3 What do we need to have a formal User Interface development method to design the 3DUI?

In terms of its formalization we need to consider:

- *Models* (Chapter 3): Conceptualization is needed in order to understand a domain, to construct artefacts and to transfer knowledge between humans and systems. A formal underpinning for describing models in a set of meta-models facilitates meaningful integration and transformation among models, and is the basis for automation through software. Models are represented as a class diagram that then can be specified using the selected language.
- *Language* (Chapter 5): To express these models a User Interface Description Language (UIDL) were chosen UsiXML, which stands for USeR Interface eXtensible Markup Language. UsiXML has been selected for our work among other reasons because:
 - *Open*: It means everybody can have access to it. UsiXML is publicly available without cost.
 - *Extensible*: In order to introduce an extension in other UIDL languages, a long process must be followed, that is not necessarily successful. Due to the fact that UsiXML is controlled by BCH any new extension to the language has been considered as an official concept.
- *Approach* (Chapter 4): MDA has been applied to many kinds of business problems and integrated with a wide array of other common computing technologies, including the area of UIs. In MDA, a systematic method is recommended to drive the development life cycle to guarantee some form of quality of the resulting software system [Vand05].
- *Tool support* (Chapter 6): Software systems can be organized around a set of models by applying a series of transformations between models, organized into an architectural framework of layers and transformations: model-to-model transformations support any change between models while model-to-code transformation are typically associated with code production, automated or not [Vand05].

7.2.4 Requirements evaluation

The internal validation of a methodology consists in assessing its characteristics against a set of selected criteria. The relevant criteria, called requirements, for our methodology have been elicited and motivated after the state of the art of Chapter 2. This section proposes a discussion for each of these requirements. A subjective

interpretation of the personal interpretation of the level of accomplishment of the different requirements is depicted in Figure 7-46. Notice that maximum value is 9 as personally I do not believe in perfection so nine means we did the maximum we were able to do.

Eight were set to those requirements from which an external opinion is needed, for instance, although we did as much as we could to produce an ontology expressive and human readable any designer might not understand it. Seven was set to the language requirement due to the limited existence of standards on which rely, although we did rely on recommendations from different recognized organizations. The methodological explicitness was set to 6 because to prove it more than any other requirement an external evaluation was needed. Finally, the software tools interoperability exists but not in an integrated development environment.

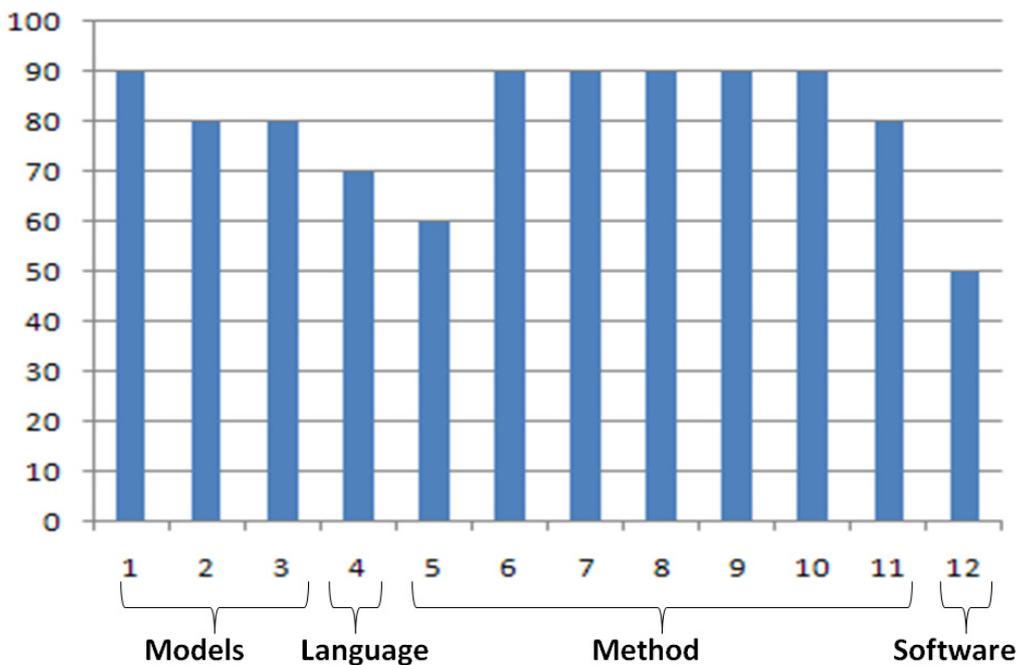


Figure 7-46. Personal subjective requirements evaluation.

7.2.4.a Evaluation of Ontological Requirements

Requirement #1: The ontology must be extensible (Shortcomings #1 and #8). It has been illustrated this requirement, particularly in the case study 3 in chapter 7, when new concepts to the CUI model were added to describe AHMI objects. The introduction of the haptic channel in the case study 4 and that were discussed in the chapter 3 shows this capability. Although, still we are not able to assure that any modality or extension to a different modality of interaction is supported, we are well aware that very complex interactions cannot be supported.

Requirement #2: The ontology must be expressive (Shortcoming #7). In this thesis, in chapter 3 our models provide enough details to allow an implementation of the system they describe. This has been reinforced through the case studies chapter.

Requirement #3: The ontology must be human readable (Shortcoming #9). The ontology presented in chapter 3 and that has been reported in different publications [Gonz09a, Kakl08a, Kakl08b, Huma08a] showing its expressivity and its human readability.

Requirement #4: The ontology should rely on standards (Shortcoming #14). Due to the lack of standards in this area this requirement is not accomplished. However, we consider recommendations from different organization, such as W3C, OMG, Web3D, that at least are followed by many researchers.

7.2.4.b Methodological Requirements

Requirement #5: Methodological explicitness (Shortcoming #2). This requirement is hard to be self-evaluated. Evidently, the steps of our methodology were defined in a way, at least we aimed, and that facilitates the comprehension of its internal logic and its application.

Requirement #6: Methodological support for multiple solutions (Shortcoming #3). This requirement is completely fulfilled and illustrated in the case studies. From one task model there are more than one FUI in case study 1 (not just final language target but the presentation is different as well). In case study 2, the multiple options for a 3D widget are shown with the use of 6 different presentation of a slider.

Requirement #7: Methodological support for homogeneity (Shortcoming #7). The methodological steps use a common semantics, expressed in UML class diagrams. The semantics are mapped to a UIDL semantics which is a XML schema. Then the UIDL proposed has XML syntax for all the different viewpoints of the methodology, including transformations.

Requirement #8: Methodological reuse (Shortcoming #4). The set of guidelines proposed along with the methodology is targeting this requirement. Patterns are a perfect match to this requirement as they promote the reuse of task models.

Requirement #9: Methodological support for the development life-cycle of 3DUI (Shortcoming #4). The development life-cycle of 3DUI is covered by the proposed methodology. It start from a task and concepts definition that is first transformed into an abstract user interface model (modality of interaction inde-

pendent model); this model is then reified into a 3DUI concrete user interface model (platform independent model); that then can be rendered or interpreted.

Requirement #10: The method must be user-centred (Shortcomings #5, #10, #11, #12). The user activities are model using a task model. This task model is then object of transformations that end in the 3DUI. Usability guidelines are considered and recommended in the mast steps of the methodology reinforcing the need to centre the development on the user.

Requirement #11: The methodology must be structured and based on principles (Shortcoming #7). The method is structured using an accepted framework (Cameleon [Calv03]). The set of principles that are along the method were introduced to accomplish this requirement. By principles it is understand: task patterns, methodological guidelines, canonical task types, usability guidelines.

Requirement #12: Support for tool interoperability (Shortcoming #14). This requirement is partially supported. Although transformation tools can be used to support the transformation from one model to another, the input an output models are not fully compatible to the modelling tools used in this dissertation.

7.3 External Evaluation

No external evaluation was conducted to effectively evaluate the impact of the methodology for several reasons:

- As announced in the introduction, this thesis was not aimed at creating a better or different methodology that could be compared with others. Rather, its goal was to provide a methodological support encouraging designers to explore design options in a principle-based way while modelling 3DUIs.
- Evaluating the FUI was not in the scope neither since the goal was not to prove that the results provided by the method are more usable or preferable compared to other similar solutions.
- If the goal was indeed to prove that the method is better than another one, proving this assertion through experiments would be more than prohibitive: the method proposed in this thesis should be compared against at least 3 or 4 different methods that should be comparable in aims and goals, the method should be applied by a significantly large amount of 3D designers and developers (who are hard to find and very expensive to pay for conducting such an experiment due to their expertise) on a significantly large amount of case studies.

Chapter 7. Validation

Two tests were conducted anyway in order to obtain some approximation of how people would perceive and estimate the benefits of the method: an informal evaluation with the haptic browser and a questionnaire to some designers.

Shortcomings reported during testing	Confrontation
Slow loading	An option was added for disabling the downloading of the images included in each web page. The alternate text is the only information regarding images that blind users can receive using the proposed rendering engine, so if this option is enabled, the loading becomes slower without providing any extra functionality.
Difficult recognition of some haptets.	A list of available haptets was created which allows user to choose the haptets that suit best to specific needs/preferences.
Some users showed special interest for some component types and complete unconcern for some others.	A new feature was added that allows user to choose which components will be used and which will be totally excluded from rendering.
Earcons may become annoying.	An option for enabling/disabling earcons was added.
The identification of the haptets was quite difficult due to similarities of their haptic attributes.	The value of friction and stiffness of each haptic became customizable.
Novice users could not easily identify the haptets due to their small size.	The size of the haptets became customizable.
The speech recognition system had a high fault percentage.	A new feature was added that allows user to type the new URL using the keyboard. Each time a key is being pressed, its value is being heard via the speech synthesis engine.
Some users could not easily move upper/lower on the page by rotating the stylus of the Phantom.	A new way of moving upper/lower on the page was offered by using the UP and DOWN keys of the keyboard.
There was no way to stop speech synthesis engine preferably. When the user was in contact with a haptic and chose to listen its value, he/she had to wait until all the content had been heard. This was annoying for haptets representing large blocks of text.	The application was extended in a way that allow user to stop the speech synthesis engine by pressing the LSHIFT key of the keyboard.

Table 7-4. Shortcomings of the rendering engine and how they have been confronted.

Haptic browser. In order to evaluate the usability of the haptic rendering engine, an informal, qualitative analysis with representative blind users from the “Thessaloniki School for the Blind” was performed. This survey provided feedback on the features of the proposed rendering engine which was used for improvement and optimization. Table 7-4 below depicts the reported shortcomings and the extensions made to confront them. The users participated in the survey were experienced on navigating through the internet using various screen readers. Comparison between navigation times using the proposed rendering engine instead of us-

ing a screen reader initially revealed that the navigation time is quicker using a screen reader. However, all blind users agreed that the proposed rendering engine has some significant advantages against the existing technologies. First of all, the user may freely navigate within the 3D scene (the pointer may asynchronously move from one object to another), while screen readers suffer from sequential navigation. Another substantial advantage of the rendering engine is that it helps blind users to have a perception of the structure of the web page that is very close to the real one (it cannot be exactly the same because 3D rendering puts some limitations in positioning) and this results to a more amusing/interesting navigation. Furthermore, it is expected that blind users will be able to reduce navigation times using the proposed rendering engine and even achieve a quicker navigation rather than using a screen reader when they will become acquainted with the proposed, novel system.

Questionnaire-based approach for designers. Seven 3DUI designers and developers were asked to express how they would perceive the benefits of the proposed methodology. Rather than applying the methodology (which would have induced a lot of time), they were told to look at the method summary available at <http://www.usixml.org/index.php?mod=pages&id=41>, where the various steps of the methods are briefly explained and exemplified by videos. If additional information was requested, the written material was made available to them, particularly the chapter where the method is summarized [Gonz09]. The anonymous designers belong to a Belgian web site agency and a Spanish School of Computer Science. The first develops 3DUIs for the web while the second is teaching 3DUI. After viewing the material, designers were asked to fill in as honestly as possible the IBM Computer Satisfaction Usability Questionnaire (CSUQ). This questionnaire was selected because of its high reliability, its simplicity, and its high correlation with the results (empirically proved with $r=0.94$) [Lew95]. This questionnaire is decomposed into 19 questions that are structured in four groups: system use (SYSUSE-Q1 to 8), information quality (INFOQUAL-Q9 to 15), interface quality (INTERQUAL-Q16 to 18), and overall estimation (OVERALL-Q19). Each question is answered on a 5-point Likert scale (Table 7-5).

Chapter 7. Validation

Question ID	Question statement	D1	D2	D3	D4	D5	D6	D7
1	Overall, I am satisfied with how easy it is to use this system	5	3	4	4	5	5	4
2	It was simple to use this system	3	4	3	3	4	3	5
3	I can effectively complete my work using this system	6	3	3	3	5	5	4
4	I am able to complete my work quickly using this system	4	3	4	4	5	3	5
5	I am able to efficiently complete my work using this system	3	4	4	4	5	4	4
6	I feel comfortable using this system	4	4	5	3	4	4	6
7	It was easy to learn to use this system	3	4	3	5	3	4	4
8	I believe I became productive quickly using this system	4	5	3	4	4	3	5
9	The system gives error messages that clearly tell me how to fix problems	4	2	3	3	4	4	5
10	Whenever I make a mistake using the system, I recover easily and quickly	5	3	3	2	4	4	4
11	The information (such as online help, on-screen messages, and other documentati	6	2	3	2	3	3	3
12	It is easy to find the information I needed	5	6	5	5	5	4	6
13	The information provided for the system is easy to understand	4	5	5	4	6	5	5
14	The information is effective in helping me complete the tasks and scenarios	4	4	5	4	5	5	5
15	The organization of information on the system screens is clear	4	5	5	5	5	5	6
16	The interface of this system is pleasant	5	5	5	5	5	5	5
17	I like using the interface of this system	6	5	5	5	6	5	6
18	This system has all the functions and capabilities I expect it to have	4	4	4	3	5	4	4
19	Overall, I am satisfied with this system	6	4	5	4	6	5	6

Table 7-5. Results from the IBM CSUQ questionnaire

	Per question statistics			
	Mean	Median	Average of deviation:	Standard deviation
Q1	4,29	4,00	0,61	0,76
Q2	3,57	3,00	0,65	0,79
Q3	4,14	4,00	1,02	1,21
Q4	4,00	4,00	0,57	0,82
Q5	4,00	4,00	0,29	0,58
Q6	4,29	4,00	0,69	0,95
Q7	3,71	4,00	0,61	0,76
Q8	4,00	4,00	0,57	0,82
Q9	3,57	4,00	0,78	0,98
Q10	3,57	4,00	0,78	0,98
Q11	3,14	3,00	0,82	1,35
Q12	5,14	5,00	0,49	0,69
Q13	4,86	5,00	0,49	0,69
Q14	4,57	5,00	0,49	0,53
Q15	5,00	5,00	0,29	0,58
Q16	5,00	5,00	0,00	0,00
Q17	5,43	5,00	0,49	0,53
Q18	4,00	4,00	0,29	0,58
Q19	5,14	5,00	0,73	0,90

Table 7-6. Statistics computed to all IBM CSUQ questions.

Although the amount of tested participants (7) is not considered as statistically significant, Table 7-6 certainly gives some indication where the benefits are potentially perceived by designers: the worst score is related to documentation (Q11 - Avg = 3,14), which is understandable since designers were not asked to run the methodology themselves but see how it can be run; the method was not judged very easy at first glance (Q12 - Avg = 3,57, same for Q9 and Q10), thus impacting the perceived easyness Q7 - Avg = 3,71). The best score was obtained with the system interface (Q17 - Avg = 5,43), which may suggest that designers tend to

perceive more the benefits of the methods through the software tools that support the method than the method itself. The structure was perceived good (Q15 - Avg = 5), and so was the pleasure to use the system (Q16 - Avg = 5). This does not necessarily mean that the method is actually structured and pleasant to use, but that it is perceived subjectively by designers like that, which seems important to identify. Although several individual questions were ranked at a medium stage, the overall satisfaction was perceived higher (Q19 - Avg = 5,14), which largely contrasts with some specific issues such as: method difficult to use, information not obviously available, but method that produces acceptable results and in an organized way. This seems to be confirmed with the four respective global scores produced by the IBM CSUQ (Fig. 7-47): the system usage is not estimated easy to use nor the information quality, but the interface quality was estimated better (with smaller variations) and the overall quality.

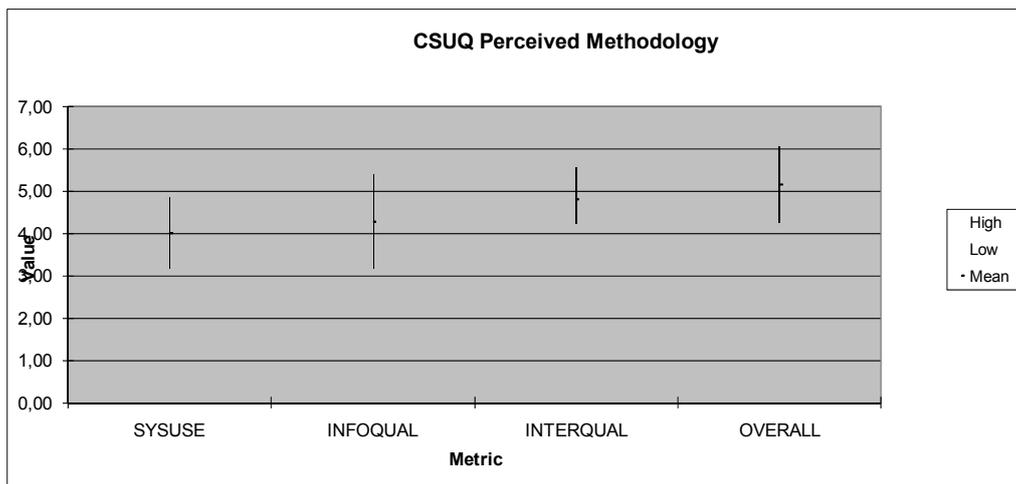


Figure 7-47. IBM CSUQ metrics.

7.4 Conclusion

In this chapter, we introduced two studies in order to show the feasibility of the method proposed for the development of 3DUIs. The two case studies show how model-based development applies to two low-complexity examples.

In order to solve these case studies we have followed the following procedure:

- (1) Building initial models. Such models have been edited with their associated editing tool.
- (2) Editing and debugging of rules within the AGG.
- (3) Importing initial models into the AGG graphical environment.
- (4) Selecting a transformation set and firing the rules contained in this set.

Chapter 7. Validation

- (5) Generate UsiXML specifications
- (6) Generate 3DUIs.

This process led us to deduce the following conclusions regarding the strengths and weaknesses of our method.

Our case study showed the **feasibility** of developing a 3DUI in a model-based relying on explicit transformation catalogs at any time. The diversity of development paths that have been presented highlight the possibility of manipulating 3DUIs related artefacts according to different development scenarios and pave the way to consider multiple other alternatives. The reuse of transformations and components has been illustrated when transformation systems have been straightforwardly reused from one case study to another one.

The feasibility of the approach is much depending on the amount and the quality of the design rules that are also encoded in UsiXML. If a reasonably extensive set of rules is used, the generated results are usable. If this is not the case, the model resulting from the transformation could be considered as underspecified. It is then required to manually edit within a XML-compliant editor. Future work will therefore be dedicated to exploring more design options and encode them in UsiXML so as to serve better transformations. This does not mean that a generated 3DUI is as usable as or more usable than a manually-produced one, but at least it could be obtained in a very fast way. Moreover, the exploration of alternative design options could be facilitated since they are operated at a higher level of abstraction than the code level.

Chapter 8 Conclusion

8.1 Contents of This Dissertation

The state of the art of Chapter 2 revealed a series of shortcomings in existing approaches for achieving transformational development of 3DUIs. Also the languages, toolkits, and render engines were examined. The shortcomings delineated our problem space. These shortcomings lead us to conclude that we can be improved the way 3DUIs is done along several dimensions.

Chapters 3, 4, 5, and 6 provide the solution to the shortcoming identified in chapter 2. For this purpose, this dissertation proposes (1) an ontological framework extending the actual description that considers just 2D UIs (2) a methodological framework, called *model-based development*, based on the ontological framework previously introduced. This methodological framework introduces a new paradigm for 3DUI development.

This development method decomposes any development activity (i.e., a development scenario) in a series of *development steps* consisting in the transformation of the artefact(s) in the scope of a *development stage* (here referred as *viewpoint*) into other development artefacts. In this context, a *development path* is defined as an archetypal composition of development steps. We identified two development paths: forward engineering, reverse engineering, even that we just show how to achieve one of those.

Chapter 7 introduced two case studies that showed the feasibility of developing a 3DUI in a model-based relying on explicit transformation catalogs at any time. The process adopted to develop the case studies of this chapter consists of: (1) Building initial models. Such models have been edited with their associated editing tool. For instance, IdealXML [Mont04] has been used to edit the task and domain model. Most of the rules have been elicited prior to realizing these case studies by a theoretical analysis of development sub-steps as illustrated in Chapter 4. (2) Exporting resulting models to UsiXML and illustration.

The first case study was devoted to the development of an opinion polling system, a reasonable scaled example of a typical information system. The purpose of this case study is to show the feasibility of solving this problem of generating 3D user interfaces by applying a model-based approach. A forward engineering path is ap-

plied that starts from the definition of the task and domain models to produce both an AUI and a CUI. The CUI is reshuffled by hand in Maya editor; these modifications do not change the AUI.

The second case study, an ongoing work, consisted in a e-Commerce application, a more complex problem not just from the point of view of implementation but also in terms of implications of using 3D interaction as an option for ecommerce. We consider this case study to show the applicability of the method on the organizational domain. Online 3D virtual spaces, games sales, and communities exist since 1996. However today we are living the era where companies are interested in being present in current virtual online applications. Our goal is to show how starting from organizational models a corresponding 3D application is produced. The method contributes to organizations to be present as a 3D organization. Notice that, in principle, one simple specification could derive several different implementations, i.e., being present in different 3D render engines.

8.2 Summary of Contributions

The development methodology relies on three main axes: models, method and language. Over the three axes we did some contributions that are summarized in the next paragraphs. The contribution to *UsiXML models* is summarized in Table 8-1. More than 200 attributes, 90 classes, 100 relationships were added to the UsiXML models that corresponds mainly to aspects for 3DUIs and some more to task modelling concepts. To reinforce or extend the models and provide substantial information to support the transformational approach these changes were made. A sanity check of the resulting models was made to consolidate them.

- *Task*: Canonical list of task types and guidelines for task modelling.
- *Domain Model*: Improving existing Domain Model that will be considered for the NexOF-RA the European standardization process.
- *AUI*: Improving existing AUI Model that will be considered for the NexOF-RA The European standardization process
- *CUI*: Enriching CUI Model by adding haptic support, separation of concerns 2D vs. 3D by specialization. 3D support. Hapgets (set of 3D widgets enhanced with haptic feedback).

The *method* aspects adhere to the MDE paradigm. Models and transformations are explicitly defined and used, around 85 transformations rules (25 old, 15 adapted, 45 new). The method relies on the Cameleon reference framework then is said that is structured. It just provides means for forward engineering. A set of princi-

Chapter 8. Conclusion

ples decomposed in: guidelines for the different development steps for evaluating the resulting models, task models relying on patterns and a canonical list of task type. All these principles promote systematicity when modelling 3DUIs.

The development steps reinforce existing knowledge on 3DUI development methods (Table 8-2) at different levels. In terms of transformational explicitness, abstraction layers independent of the modality of interaction. We provide some means to identify the diversity of concretizations of the 3DUI (i.e. its representation) for a task.

The summary of the development steps is:

- *Step 1: Task and concepts (T&C) definition*, including: Task and Concepts Consolidation: task and domains concepts are object of transformation rules in order to have mappings from domain concepts and tasks; Selection of task types, items and user categories for each task; and Guidelines of best-practices for structuring task models.
- *Step 2: From task and Concepts to Abstract UI (AUI) Model*, including: Identification of abstract UI structure; Selection of Abstract Interactive Components; Spatio-Temporal arrangement of abstract interaction objects; Definition of abstract dialog control; and Derivation of AUI to domain mappings.
- *Step 3: From AUI to CUI Model*, including: Reification of Abstract Interactive Components into Concrete 3D elements; Selection of 3D Concrete Interactive Components; and Selection of the graphical representation of the 3D Concrete Interactive Components.
- *Step 4: From CUI Model to FUI*.

Finally, the *evaluation of 3DUIs* is achieved by relying on guidelines (Table 8-3). Guidelines were extracted from the literature and adapted to our needs. Some new were added and some more could be incorporated particularly for evaluating the Final User Interface. Still any other traditional method (see list in Table 8-3) can be used to evaluate the user interface. Relying on guidelines is just a low cost option in terms of budget and availability of users for the tests, particularly when intending to conduct a usability test based on experts views.

The *language*, all aspects are stored in UsiXML files that can be exchanged, shared, and communicated between stakeholders (designers, developers, and end users). Advantages of this language were discussed, including: modifiability, complexity, rigorousness, allows reasoning, is processable. UsiXML is a language that adopts a Language Engineering Approach because it relies on three axes: semantics ex-

pressed as Meta Models using UML class diagrams; syntax: abstract as a XML schema and concrete XML; stylistics providing different graphical representations of the concepts of the different development steps. One important aspect of the language proposed is that our contribution is independent of UsiXML. The knowledge versed in this work can be use in any other domain.

The *software support* for the method plays an important role. In our thesis we use five Software modules are used to support the methodology. We participate in the development of such software modules by participating in their conceptual definition, supervising their development, testing with our case studies the performance. Finally, the final user interface rendering corresponds to more than 10, 000 LOC.

8.3 On the promises of MDA

Usually, we can say that there a set of expected benefits of Model-Driven Engineering (MDE). In this dissertation we did follow this approach for our method but not all the benefits are fully observable. In details what it was expected and what we got can be subsumed as follows.

Benefits resulting from the existence of a design phase. Reducing the gap between requirements and implementation: A design phase aims to ensure in advance that the implementation really addresses the customer's and user's requirements captured in our case in the task model. Developer coordination: Previous planning of the developed system enables the developers to coordinate their work e.g. by dividing the system into several parts (models) and defining interfaces between them (mappings). It is expected to end with well-structured systems, which includes a design phase providing explicit planning of the system architecture and the overall code structure. This facilitates implementation itself as well as maintenance. Overall this added value can be found in our work.

On the benefits resulting from the use of visual abstract models we benefit from planning on adequate level of abstraction (task, AUI, CUI models). Modelling languages provide the developer concepts for planning and reasoning about the developed system on an adequate level of abstraction. It is expected an improvement in the communication by visual models (stylistics of the language): The visual character of modelling languages can lead to increased usability (understanding, perceiving, exploring, etc.,) of design documents for both author and other developers, this aspect for sure needs further validation against the usability evaluation with developers and designers. Validation: (Semi-) Formal modelling languages enable automatic validation of the design. Models can be used as documentation when maintaining the system saving time for the documentation. The

3D CUI model can be reused or at least serve as starting point when implementing the system for a different platform. This includes development platforms like a programming language or component model, as well as deployment platforms like the operating system or target devices.

Benefits resulting from code generation are expected on an enhancement in productivity. Generating code from a given model requires often only a teeny part of time compared to manual mapping into code, this aspect need a more elaborated evaluation comparing development time with and without the use of the proposed method. We did add our knowledge into the code generator. Without consider our self experts but competent in this field, the benefit of doing this contributes on code structuring, code optimizations, or platform-specific features – can once be put into the code generator and then be reused by all developers. Finally, automatic code generation should reduction of errors, automatic mapping prevents from manual errors.

Benefits of meta-modelling include the easy creation and maintenance of development support. Little support was the result for an integrated development environment then we did not fully benefit to provide sophisticated support for all steps in MDE like creating and processing metamodels, creating modelling editors, and defining and executing transformations. On the other hand, our systematic and explicit definition of metamodels and transformations facilitates maintenance of modelling languages and code generators; this can be done on the models and language. Ultimately, our MDE compliant explicit metamodels and transformations can easily be understood and reused by others.

8.4 Future work

The future work consists of:

- *Extending the set of 3DUIs components.* We have some implementation that we have present in the corpus of this text. However, we need not just to have the implementation but also explore more genuine and innovative representations.
- *Extending the set of rules.* The set of rules that we exposed require a transformation engine that support the whole path, currently the tool is being developed. The remaining tool will be dedicated to transform CUI model to a high level virtual content editor.
- *Re-expression of usability knowledge in terms of 3DUI concepts.* Considering design evaluation strategies based on the concepts managed by the method-

Chapter 8. Conclusion

ology. It can be more easily to identify ergonomic rules, at a high level of the methodological path than on code.

- *Scalability of the method.* We illustrated the principles of model-based transformational development for one case study. While this case study is intended to demonstrate the feasibility of our research, the second one is under the scope of the organization. We will conclude a second case study related to a virtual e-shop.
- Appendixes C (3DUI representations), and D (Task Types) contains on-going research which is not yet finished.
- *Evaluation of the 3DUI representation* might have some impact in future research. We wonder if the 3DUI representation has an impact on user interaction. For instance, considering the slider, and the different representation we proposed, consider a cone and a cylinder that has their lowest value on the base and the highest on the top. While the cylinder has a constant shape from bottom to the top. The cone is bigger on the bottom and thinner on the top. Would a user have tendency to select an increasing value similarly in a cone compared to a cylinder?
- There is some concrete interest to apply this method for a 3D representation of learning objects [Gonz09b]. For sure, a 3D representation of a learning object might be out of the scope of this research as it could be composed of animations but the controls to manipulate such animations will be generated with our methodology.
- There was interest to use our 3D widgets for augmented reality application in an intelligent kitchen. Even that the project finally was not consolidated, the idea showed the opportunity to explore other Virtual reality areas that were not explored in this thesis.

3DUIs are not intended for everybody. For the increasing number of users interested by 3DUIs, organizations need to be present in virtual online applications. For those users who are not interested by 3DUIs, our method benefits from using the MDA approach introduced in this thesis that supports multi-path development, i.e., one source, many targets, an important benefit [Kels03]. These targets vary in computer platforms (e.g., mobile phone, desktop), programming languages (e.g., Java, Html). The goal of this thesis was not to prove that the method is better than another, neither the usability of the 3DUI produced. The goal was to define a method to develop 3DUI in a principle-based way as opposed to an opportunistic way. Therefore, the appropriateness of this method in this dissertation is for sure the major piece of work to investigate in the near future.

Chapter 8. Conclusion

UsiXML	Task and Concepts	Abstract User Interface	Concrete User Interface	Final User Interface	Context
Old	<p>Task = extended CTT, based on Markopoulos LOTOS desc.</p> <p>Domain = UML class diagram + extensions in a profile</p>	<p>AUI = hierarchy of abstract containers (ACs) and Abs. Indiv.</p> <p>Comp. (AICs) and relations</p> <p>AIC = faceted computing: input, output, control, navigation</p> <p>Relations = structural, temporal</p>	<p>CUI = hierarchy of concrete interaction objects (CIOs) + behaviour</p> <p>CIO = graphical / auditory</p> <p>Graphical CIO = containers (window, dialog box,...) or indiv. (check box)</p> <p>Auditory CIO = form, group, field, value (VoiceXML)</p> <p>Behaviour = set of ECA rules (events, conditions, actions)</p>	<p>Markup: Flash, WML, XHTML, X+V</p> <p>Programming: C++, Java,</p>	<p>User population = hierarchy of user stereotypes with param.</p> <p>Platform = subset of CC/PP (UAProf)</p> <p>Environment = physical, psychological, organisat. properties</p>
New	<p>Canonical list of action types</p> <p>DomainModel= Submodel + Element + Collection + Items.</p> <p>Domain relationships: Existance, relevance, cardinality, replication</p>	<p>AUI = AbstractInteractionUnits and abstract behaviour</p> <p>AuiObjects = AuiInteractor: Data Interactor and AuiContainer. Including relationships (hierarchy and grouping) among them.</p> <p>Data Interactor: input, output, selection.</p> <p>Trigger interactor: command navigator.</p> <p>AbstractBehaviour: canonical list of events, condition, canonical list of actions.</p>	<p>Hapget = 3D CIO augmented with haptic parameters</p> <p>CIO = 3D / hapget</p>	<p>Java3D, VRML, X3D</p>	

Table 8-1. Contribution to UsiXML ontology.

Chapter 8. Conclusion

	Models t= task, Do = Domain Di = dialog AUI=abstract presentation CUI=concrete user interface U = user, C = context.	Inter Model Transformation ↔Bidirectional derivation → Derivation link ≈→ Manual Derivation ≈ ↔Manual Bidirectional der. FUI = Final User Interface	FUI target languages	Avl Mod.	Ext. Mod
VR-Wise	CUI	CUI → FUI	VRML, X3D	√	Orig.
CoGenIVE	T, Di, CUI	(T, Di, CUI) ↔ FUI	C++	√	Orig.
InTML	Di, CUI	T ≈→ (Di, CUI), (Di, CUI) ≈→ FUI	VRML	√	Orig.
Contigra	CUI, Di	(CUI, Di) → FUI	X3D, Behavior3D, Audio3D	√	Orig.
Tres-D	T, U, C, Di, Do	T, Do ≈→ CUI, CUI ≈→ FUI	X3D, VRML	√	Design
Animation	T, CUI, Di	T ≈→ CUI, CUI → FUI	C	x	x
Desktop 3DUIs	T, CUI, Di	T ≈→ CUI, CUI → FUI	X3D, VRML	√	Design
Participative	T, CUI, Di	T ≈→ CUI, CUI ≈→ FUI	Not specified	√	Orig.
Task analysis	T, CUI, Di	T ≈→ CUI	None	√	Orig.
This thesis	T, Do, Di, C, AUI, CUI	T ↔ Do, (T, Do, C) ↔ AUI, AUI ↔ CUI, CUI ↔ AUI, T ↔ CUI, (T, Do, AUI, CUI) ↔ C, CUI ≈→ FUI	Java, XHTML, Flash, HTML, Voice XML, Java 3D, X3D, VRML.	√	Orig.

Table 8-2. Model-based methodologies contribution.

Chapter 8. Conclusion

	Task & Concepts	Usability Evaluation Method		
		AUI	CUI	FUI
[Wils02]	(M) Feedback definition, (B) Task analysis (M) Interaction techniques selection based on survey of available interactions.	----	----	(M) Fidelity and validity of the virtual environment.
[Cele01]	----	----	----	(M) User and Expert experience evaluation
[Gabb99]	(B) Task analysis (M) Guideline-based expert evaluation	----	(M) Heuristic evaluation based on guidelines expert evaluation on prototypes	(M) Experts guidelines evaluated on the VE
[Nede06]	(M) Usability aspects to be evaluated are defined	----	(M) Design User experiments	(M) User experience evaluation based on questionnaires
[Neal01]	----	----	----	(M) Evaluation on user and expert experience
[Kaur98]	(M) Guideline-based evaluation	----	(M) Guideline-based evaluation	(M) Guideline-based evaluation
[Hack98]	(B) Task analysis	----	----	----
[Witm98] [Kenn93]	---	---	----	(M) Presence Questionnaire (M) Simulator sickness questionnaire
[Bowm04]	(M) User task (task properties) scenarios. (M) Interaction Technique evaluation based on taxonomy decomposition, (M) Context (environment, user, system) analysis, (B) Summative evaluations of IT	----	(M) Evaluation on low-fidelity prototyping. (M) Wizard of OZ.	Summative evaluation (M) Testbed evaluation
[Hix93]	(B) Summative evaluations of IT	----	(M) Formative evaluation with Prototyping	----
[Card90]	(B) Multidimensional design spaces	----	----	----
[Poup97]	Metaphor-based classifications	----	----	Summative evaluation
[Stee98]	----	----	(M) Guideline-based expert evaluation on prototypes	Cognitive Walkthrough
This thesis	(B) Evaluation based on guideline rules			

Table 8-3. Coverage of the evaluation method proposed.

References

A

- [Abra99] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S., Shuster, J., *UIML: An appliance-independent XML user interface language*, (Toronto, 11-14 May 1999), Elsevier Science Publishers, Amsterdam, 1999, pp. 661–665.
- [Alex77] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S., *A Pattern Language*, Oxford University Press, New York, 1977.
- [Anar06a] *Products: Anark Studio*, Anark Corporation, 2009. Available on-line at: http://www.anark.com/products/products_studio.asp
- [Anar06b] *Anark Gameface Evaluation Supplement*, Anark Corporation, 2009. Available on-line at: <http://update.anark.com/Gameface/EvaluationSupplement.html#examples>
- [Andu06] Andujar, C., Fairén, M., Argelaguet, F., *Cost Effective Approach for Developing Application-Control GUIs for Virtual Environments*, Proc. of the 1st IEEE Symposium of 3D User Interfaces 3DUI'2006 (Alexandria, 25-26 March 2006), IEEE Comp. Society Press, Los Alamitos, 2006, pp. 45–52.
- [Anne67] Annett, J., Duncan, K., *Task analysis and training design*, Occupational Psychology, Vol. 41, 1967, pp. 211-227.
- [Arsa02] Arsanjani, A., Chamberlain, D., *et al.*, (*WSXL*) *web service experience language version*, 2002. Available on-line at: <http://www-106.ibm.com/developerworks/library/ws-wsxl2/>
- [Auto06a] *Autodesk Inventor Series*, Autodesk, Inc., 2006. Available on-line at: <http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=5182857>
- [Auto06b] *Autodesk Maya*, Autodesk, Inc., 2006. Available on-line at: <http://usa.autodesk.com/adsk/servlet/index?id=6871843&siteID=123112>
- [Avaz06] Avanzini, F., Crosato, P., *Haptic-auditory rendering and perception of contact stiffness*, Proc. of 1st Int. Workshop on Haptic and Audio Interaction Design HAID'2006 (Glasgow, 31 August-1 September 2006), Lecture Notes in Computer Science, Vol. 4129, Springer, Berlin, 2006, pp. 24–31.
- [Azev00] Azevedo, P., Merrick, P., Roberts, D., *OVID to AUIML user oriented interface modelling*, Proc. of 1st Int. Workshop Towards a UML Profile for Interactive Systems Development TUPIS'00 (York, October 2000). Available on-line at: <http://xml.coverpages.org/TUPIS2000-Azevedo.html>

References

B

- [Bach04]
Bach, C., *Elaboration et validation de Critères Ergonomiques pour les Interactions Homme-Environnements Virtuels*, Ph.D. Thesis, Université de Metz, Metz, 2004.
- [Bake06a]
Baker, M.J., *EuclideanSpace - building a 3D world*, 2006. Available on-line at: <http://www.euclideanpace.com/>
- [Bake06b]
Baker, D., *The database for freeware Max R8 plug-ins*, 2006. Available on-line at: <http://www.maxplugins.de/max8.php>
- [Balz95]
Balzert, H., *From OOA to GUI - The JANUS-System*, Proc. of the 5th IFIP TC13 Conference on Human-Computer Interaction INTERACT'95 (Lillehammer, 25-29 June 1995), Chapman & Hall, London, 1995, pp. 319–324.
- [Barb95]
Barboni, E., Navarre, D., Palanque P., Basnyat, S., *Exploitation of Formal Specification Techniques for ARINC 661 Interactive Cockpit Applications*, Proc. of Int. Conf. HCI Aero2006 (Seattle, September 2006), pp. 81–89.
- [Bart88]
Barthet, M.-F., *Logiciels interactifs et ergonomie*, Dunod Informatique, Paris, 1988.
- [BBCN07a]
Waters, D., *Virtual Worlds set for shake-up*, Story from BBC NEWS, 8 March 2007. Available on-line at: <http://news.bbc.co.uk/go/pr/fr/-/2/hi/technology/6431207.stm>.
- [BBCN07b]
Virtual worlds are worth \$1bn, Story from BBC NEWS, 20 March 2007. Available on-line at: <http://news.bbc.co.uk/go/pr/fr/-/2/hi/technology/6470433.stm>.
- [Beau00]
Beaudouin-Lafon, M., *Instrumental Interaction: an Interaction Model for Designing Post-WIMP User Interfaces*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2000 (La Haye, April 2000), ACM Press, New York, 2000, pp.446-453.
- [Bier01]
Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C., *VR Juggler: A Virtual Platform for Virtual Reality Application Development*, Proc. of IEEE Int. Conf. on Virtual Reality VR'2001 (Yokohama, 13-17 March 2001), IEEE Comp. Society Press, Los Alamitos, 2001, pp. 89–96.
- [Bimb05]
Bimber, O., Raskar, R., *Spatial Augmented Reality: Merging Real and Virtual Worlds*, AK Peters Ltd., Wellesley, 2005.
- [Bles90]
Bleser, T.W., Sibert, J., *Toto: a tool for selecting interaction techniques*, Proc. of ACM Symposium on User Interface Software and Technology UIST'90 (Snowbird, 3-5 October 1990), ACM Press, New York, 1990, pp. 135–142.
- [Boda94]
Bodart, F., Vanderdonckt, J., *On the Problem of Selecting Interaction Objects*, Proc. of BCS Conf. on Human-Computer Interaction HCI'94 "People and Computers IX" (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (eds.), Cambridge University Press, Cambridge, pp. 163–178.

References

- [Boda95]
Bodart, F., Hennebert, A., Lheureux, J., Provot, I., Sacré, B., Vanderdonckt, J., *Towards a Systematic Building of Software Architecture: The TRIDENT Methodological Guide*, Proc. of 2nd Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'95 (Toulouse, 7-9 June 1995), Ph. Palanque, R. Bastide (eds.), Springer-Verlag, Vienna, 1995, pp. 262-278..
- [Boeh88]
Boehm, B., *A Spiral Model of Software Development and Enhancement*, IEEE Communications, Vol. 21, No. 5, May 1988, pp. 61-72.
- [Boms98]
Bomsdorf, B., Szwillus, G., *From task to dialogue: Task-based user interface design*, SIGCHI Bulletin, Vol. 30, 1998, pp. 40-42.
- [Boms99]
Bomsdorf, B., Szwillus, G., *Tool support for task-based user interface design*, SIGCHI Bulletin, Vol. 31, No. 4, 1999, pp. 27-29.
- [Bori97]
Boritz, J., Booth, k., *A study of interactive 3D point location in a computer simulated virtual environment*, Proc. of ACM Conf. on Virtual Reality Software Technology VRST'97 (Lausanne, 15-17 September 1997), ACM Press, New York, 1997, pp. 181-187.
- [Bosw02]
Boswell, D., King, B., Oeschger, I., Collins, P., Murphy, E., *Introduction to XUL*. In "Creating Applications with Mozilla", O'Reilly, Sebastopol, September 2002.
- [Boui04]
Bouillon, L., Vanderdonckt, J., Chow, K.C., *Flexible Re-engineering of Web Sites*, Proc. of 8th ACM Int. Conf. on Intelligent User Interfaces IUI'2004 (Funchal, 13-16 January 2004), ACM Press, New York, 2004, pp. 132-139.
- [Bowm97a]
Bowman, D., Hodges, L., *An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments*, Proc. of the 1997 Symposium on Interactive 3D Graphics, SI3D '97 (Providence, 27-30 April 1997), ACM Press, New York, 1997, pp. 35-38.
- [Bowm97b]
Bowman, D., Koller, D., Hodges L., *Travel in Immersive Virtual Environments: an Evaluation of Viewpoint Motion Control Techniques*, Proceedings of Virtual Reality Annual International Symposium (1-5 March 1997), IEEE Computer Society Press, Los Alamitos, 1997, pp. 45-52.
- [Bowm98]
Bowman D. Wineman, J., Hodges L., Allison, D., *Designing Animal Habitats within an Immersive VE*, IEEE Computer Graphics and Applications, Vol. 18, No. 5, 1998, pp. 9-13.
- [Bowm99a]
Bowman, D., Hodges, L., *Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments*, Journal of Visual Languages and Computing, Vol. 10, No. 1, 1999, pp. 37-53.
- [Bowm99b]
Bowman, D., *Interaction Techniques for Immersive Virtual Environments: Design, Evaluation, and Application*. Available on-line at: <http://people.cs.vt.edu/~bowman/papers/hcic.pdf>.

References

- [Bowm00] Bowman, D., *The Science of Interaction Design*, in Bowman, D., Kruijff, E., LaViola, J., Poupyrev, I., and Mine, M., SIGGRAPH 2000 Course 3D User Interface Design: Fundamental Techniques, Theory, and Practice, 2000.
- [Bowm01a] Bowman, D., *Basic 3D Interaction Techniques*, in Bowman, D., Kruijff, E., LaViola, J., Poupyrev, I., and Mine, M., SIGGRAPH 2001 Course Advanced Topics in 3D User Interface Design, 2001.
- [Bowm02] Bowman, D., Gabbard, J., Hix, D., *A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods*, Presence: Teleoperators and Virtual Environments, Vol. 11, No. 4, 2002, pp. 404-424.
- [Bowm04] Bowman, D.A., Kruijff, E., LaViola, J., Poupyrev, I., *3D User Interfaces: Theory and Practice*, Addison Wesley, Boston, July 2004.
- [Brew98] Brewster, S.A., *The design of sonically-enhanced widgets*, Interacting with Computers, Vol. 11, No. 2, 1998, pp. 211–235.
- [Brew07] Brewster, S., Chohan, F., Brown, L. *Tactile feedback for mobile interactions*, Proc. of ACM Conference on Human Factors in Computing Systems CHI'2007 (San Jose, 28 April-3 May 2007), ACM Press, New York, 2007, pp. 159–162.
- [Bull03] Bullard, L., *Extensible 3D: XML Meets VRML*, August 06, 2003. Available on-line at: <http://www.xml.com/pub/a/2003/08/06/x3d.html>.
- [Byrne92] Byrne, E. J., *A conceptual foundation for software re-engineering*, Proc. of the Int. Conf. on Software Maintenance (Orlando, 9-12 November 1992), IEEE Computer Society Press, Los Alamitos, 1992, pp. 216-235.
- C**
- [Calh84] Calhoun, G. C., Arbak, C. L., Boff, K. R., *Eye-controlled switching for crew station design*, Proc. of the 28th Annual Meeting of the Human Factors Society HFES'84 (Santa Monica, 1984), Human Factors Society, 1984, pp. 258-262.
- [Calv03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., *A Unifying Reference Framework for Multi-Target User Interfaces*, Interacting with Computers, Vol. 15, No. 3, 2003, pp. 289–308.
- [Card83] Card, S.K., Moran, T.P., Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, New York, 1983.
- [Card90] Card, S.K., Mackinlay, J.D., Robertson, G.G., *The design space of input devices*, Proc. of the ACM Int. Conf. on Human Factors in Computing Systems CHI'1990 (Seattle, 1-5 April 1990), ACM Press, New York, 1990, pp. 117-124.

References

- [Care97] Carey, R., Bell, G., *The annotated VRML 2.0 Reference Manual*. Available on-line at: <http://accad.osu.edu/~pgerstma/class/vnv/resources/info/AnnotatedVrmlRef/Book.html>, (1997).
- [Carn06] Carnegie Mellon University, *Alice: Learn to Program Interactive 3D Graphics*. Available on-line at: <http://www.alice.org/>
- [Carr03] Carrol, J., *HCI models, theories, and frameworks: toward a multidisciplinary science*, Morgan Kaufmann, San Francisco, 2003.
- [Cele01a] Celentano, A., Pittarello, F., *Class of Experience: A High Level Approach to Support Content Experts for the International Workshop on Authoring of 3d Environments*. Structured Design of Virtual Environments and 3d-Components, Shaker Verlag, 2001. Available on-line at: http://www.dsi.unive.it/~auce/docs/celentano_web3d01.pdf
- [Cele01b] Celentano, A., Pittarello, F., *A content centered methodology for authoring 3d interactive worlds for cultural heritage*, Proc. of Int. Cultural Heritage Informatics Meeting ICHIM'2001 (Milán, 3-7 September 2001), D Bearman, F Garzotto (Eds.), Vol. 2, Politecnico di Milano, Italia y Archives & Museum Informatics, 2001, pp. 315-324.
- [Chap05] Chapuis, O., Roussel, N. *Metisse is not a 3D desktop*, Proc. of ACM Conf. on User Interface Software Technology Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology UIST'2005 (Seattle, 23-26 October 2005), ACM Press, New York, 2005, pp. 13-22.
- [Chik90] Chikofsky, E.J., Cross, J.H., *Reverse Engineering and Design Recovery: A Taxonomy*, IEEE Software, Vol. 1, No. 7, January 1990, pp. 13-17.
- [Cler05] Clerckx, T., Luyten, K., Coninx, K., *Dynamo-AID: A Design Process and a Runtime Architecture for Dynamic Model-based User Interface Development*, Proc. of 9th IFIP Working Conf. on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, 11-13 July 2004), Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 77-95.
- [Cock00] Cockburn, A., McKenzie, B., *An evaluation of cone trees*, Proc. of the 2000 British Computer Society Conference on Human-Computer Interaction HCI'2000 (Sunderland, September 2000), Springer-Verlag, Berlin, 2000. Available on-line at: <http://eprints.kfupm.edu.sa/24597/1/24597.pdf>
- [Cock01] Cockburn, A., McKenzie. *3D or not 3D? Evaluating the Effect of the Third Dimension in a Document Management System*, Proc. of the ACM Conf. on Human factors in computing systems CHI'2001 (Seattle, 31 March-5 April 2001), ACM Press, New York, 2001, pp. 434-441. Available on-line at: <http://www.cosc.canterbury.ac.nz/~andrew.cockburn/papers/chi01DM.pdf>

References

- [Cock02]
Cockburn, A., McKenzie. *Evaluating the effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2002 (Minneapolis, 20-25 April 2002), ACM Press, new York, 2002, pp. 203-210.
- [Conn92]
Conner, D.B., Snibbe, S.S., Herndon, K.P., Robbins, D.C., Zeleznik, R.C., van Dam, A., *Three-Dimensional Widgets*, Special Issue of Computer Graphics, Proceedings of the 1992 Symposium on Interactive 3D Graphics, ACM Press, New York, 1992, pp. 183-188.
- [Cons03]
Constantine, L.L., *Canonical Abstract Prototypes for Abstract Visual and Interaction*, Proc. of the 10th Int. Workshop on Design, Specification and Evaluation of Interactive Systems DSV-IS'2003 (Funchal, 11-13 June 2003), Lecture Notes in Computer Science, Vol. 2844, Springer Verlag, Berlin, 2003, pp. 1-15.
- [Corr97]
Corradini, Ehrig, H. , Heckel, R. , Korff, M. , Löwe, M. , Ribeiro, L., Wagner A., *Algebraic approaches to graph transformation - part I: Single pushout approach and comparison with double pushout approach*, in Handbook of Graph Grammars and Computing by Graph Transformation, Rozenberg G. (Ed.), Vol. I: Foundations, World Scientific, 1997, pp. 247-312.
- [Craz05]
Crazy Eddies' wiki page. *Crazy Eddie's GUI System Namespace List*. 2005. Available on-line at: http://www.cegui.org.uk/api_reference/namespaces.html
- [Craz06]
Crazy Eddies web site. *Welcome to Crazy Eddie's GUI System*. 2006. Available on-line at: http://www.cegui.org.uk/wiki/index.php/Main_Page
- [Cube07]
Web site of Cube Productions Inc. Available on-line at: <http://cube3.com/>
- [Cupp04]
Cuppens, E., Raymaekers, Ch., Coninx, K., *Vrixml: A User Interface Description Language for Virtual Environments*, Proc. of the AVT'2004 Workshop Developing User Interfaces with XML: Advances on User Interface Description Languages UIXML 2004 (Gallipoli, May 25, 2004), 2004, pp. 111-118.
- [Cupp05a]
Cuppens, E., Coninx, K., *CoGenIVE: Code Generation for Virtual Environments*, Proc. of the ACM CHI'2005 Workshop on Future of User Interface Design Tools (Portland, 4 April 2005).
- [Cupp05b]
Cuppens, E., Raymaekers, Ch., Coninx, K., *A Model-Based Design Process for Interactive Virtual Environments*, Proc. of Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005 (Newcastle upon Tyne, 13-15 July 2005), Lecture Notes in Computer Science, Vol. 3941, Springer, Berlin, 2005, pp. 225-236.
- [Czar00]
Czarnecki, K., Eisenecker, U.W., *Generative Programming: Methods, Tools, and Applications*. In Addison-Wesley, Reading, 2000.

D

- [Dach01]
Dachselt, R., Ebert, J., *Collapsible Cylindrical Trees: A Fast Hierarchical Navigation Technique*, Proc. of the IEEE Symposium on Information Visualisation InfoVis'2001 (San Diego, 22-23 October 2001), IEEE Computer Society Press, Los Alamitos, 2001, pp. 79.

References

- [Dach02] Dachsel, R., Hinz, M., Meißner, K. *CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications*, Proc. of 7th Int. Conf. on 3D Web Technology Web3D'2002 (Tempe, 24-28 February 2002), ACM Press, New York, 2002, pp. 155–163.
- [Dach03] Dachsel, R., Rukzio, E., *BEHAVIOR3d: an xml-based framework for 3d graphics behaviour*, Proc. of the 8th Int. Conf. on 3D Web Technology Web3D'2003 (Saint Malo, 9-12 March 2003), ACM Press, New York, 2003, pp. 101–112.
- [Dach06] Dachsel, R., Figueroa, P., Lindt, I., Proc. of the IEEE Virtual Reality VR 2006 Workshop on the Specification of Mixed Reality User Interfaces: Approaches, Languages, Standardization MRUI'2006 (Alexandria, 25 March 2006). Available on-line at: <http://w3-mmt.inf.tu-dresden.de/vr2006/>
- [Dach07] Dachsel, R., Figueroa, P., Lindt, I., Broll, W., Proc. of the IEEE Virtual Reality VR 2007 Workshop on the Specification of Mixed Reality User Interfaces: Approaches, Languages, Standardization MRUI'2007 (Charlotte, 11 March 2007), Shaker Verlag, Available on-line at: <http://w3-mmt.inf.tu-dresden.de/mrui2007/>
- [Daly02] Daly, I., Brutzman, D., Hudson, A., *Introducing X3D a Tutorial*. SIGGRAPH, San Antonio, July 2002. Available on-line at: <http://www.realism.com/x3d/presentations/s2002/>
- [Davi04] Davidsson, O., et al., *Game Design Patterns for Mobile Game*, Project report to Nokia Research Centre, Finland, 2004.
- [DeBo06a] De Boeck, J., Gonzalez-Calleros, J.M., Coninx, K., Vanderdonck, J., *Open Issues for the development of 3D Multimodal Applications from an MDE perspective*, Proc. of 2nd Int. Workshop on Model Driven Development of Advanced User Interfaces MDDAU'2006 (Geneva, 2 October 2006), A. Pleuss, J. Van den Bergh, H. Hussmann, S. Sauer, A. Boedcher, (ed.), 2006, pp. 11-14.
- [DeBo06b] De Boeck, J., Raymaekers, Ch., Coninx, K., *Exploiting Proprioception to Improve Haptic Interaction in a Virtual Environment*, Presence, Vol. 15, No. 6, 2006, pp. 627-636.
- [DeBo08] De Boeck, J., Raymaekers, Ch., Coninx, K., *A Tool Supporting Model Based User Interface Design in 3D Virtual Environments*, Proc. of the 3rd Int. Conf. on Computer Graphics Theory and Applications GRAPP'2008 (Funchal, January 22-25, 2008), INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2008, pp. 367-375.
- [deLa02] de Lara, J., Vangheluwe, H., *AToM²: A tool for multi-formalism and metamodelling*, Proc. of the 5th International Conference on Fundamental Approaches to Software Engineering FASE'2002 (Grenoble, 8-12 April 2002), Lecture Notes in Computer Science, Vol. 2306, Springer-Verlag, Berlin, 2002, pp. 174-188.
- [Dela07] Delacre, J.-P., *A Comparative Analysis of Transformation Engines for User Interface Development*, M.Sc. thesis, Université catholique de Louvain, Louvain-la-Neuve, 28 August 2007. Available on-line at <http://www.isys.ucl.ac.be/bchi/publications/2007/Delacre-MSc2007.pdf>

References

- [Delé09] Deléglise, E., Paul, D., Fjeld, M., *2D/3D Web Transitions: Methods and Techniques*, Proc. of the 5th Int. Conf. on Web Information Systems and Technologies WEBIST'2009 (Lisbon, 23-26 March 2009), INSTICC Press, 2009, pp. 294-298.
- [Deml03] Demler, G., Wasmund, M., Grassel, G., Spriestersbach, A., Ziegert, T., *Flexible pagination and layouting for device independent authoring*, Proc. of WWW2003 Emerging Applications for Wireless and Mobile access Workshop.
- [Diet01] Dietrich, O., *Virtual Reality and cognitive process*, in "Virtual Reality: Cognitive Foundations, Technological Issues & Philosophical Implications", Riegler, A., Peschl, M., Edlinger, K., Fleck, G., Feigl, W. (Eds.), Peter Lang, Frankfurt am Main, 2001.
- [Dijk76] Dijkstra, E.W., *The discipline of programming*, Prentice Hall, Engelwood Cliffs, 1976.
- [Dsou99] D'Souza, D.F., Wills, A.C., *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison-Wesley, Reading, 1999.
- [Düns07] Dünser, A., Grasset, R., Seichter, H., Billingham, M., *Applying HCI Principles in AR Systems Design*, Proc. of the 2nd Int. Workshop on Mixed Reality User Interfaces: Specification, Authoring, Adaptation MRUI '07 (Charlotte, 11 March 2007).
- E**
- [East01] Eastgate, R., *The Structured Development of Virtual Environments: Enhancing Functionality and Interactivity*, Ph.D. Thesis, University of York, York, 2001.
- [Edli01] Edlinger K., *Virtual Reality, Cyberspace and Living Organisms. Towards a new understanding of perception and cognition?* in "Virtual Reality: Cognitive Foundations, Technological Issues & Philosophical Implications", Riegler, A., Peschl, M., Edlinger, K., Fleck, G., Feigl, W. (Eds.), Peter Lang, Frankfurt am Main, 2001.
- [Ehri99] Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.), *Handbook of Graph Grammars and Computing by Graph Transformation, Application, Languages and Tools*, Vol. 2, World Scientific, Singapore, 1999.
- [Eise00] Eisenstein, J., Vanderdonckt, J., Puerta, A., *Adapting to Mobile Contexts with User-Interface Modeling*, Proc. of 3rd IEEE Workshop on Mobile Computing Systems and Applications WMCSA'2000 (Monterey, 7-8 December 2000), IEEE Press, Los Alamitos, 2000, pp. 83-92.
- [Eise01] Eisenstein, J., Vanderdonckt, J., Puerta, A., *Model-based User-Interface Development Techniques for Mobile Computing*, Proc. of 5th ACM Int. Conf. on Intelligent User Interfaces IUI'2001 (Santa Fe, 14-17 January 2001), ACM Press, New York, 2001, pp. 69-76.
- [EON] Eon Reality Inc., 2009. Available on-line at: <http://www.eonreality.com/>

References

[Erme99]

Ermel, C., Rudolf, M., Taentzer, G., *The AGG-Approach: Language and Tool Environment*, in "Handbook on Graph Grammars and Computing by Graph Transformation", H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg (eds.), Vol. 2: App., Languages and Tools, World Scientific, Singapore, 1999, pp. 551–603.

F

[Fair93]

Fairchild, K.M., *Information Management Using Virtual Reality-Based Visualizations*, in "Virtual Reality: Applications and Explorations", A. Wexelblat (Ed.), Academic Press Professional, Cambridge, 1993, pp. 45-74.

[Fenc99]

Fencott, C., *Towards a Design Methodology for Virtual Environments*, Proc. of User Centered Design and Implementation of Virtual Environments (UCDIVE) Workshop, University of York, York, 30 September 1999.

[Fenc01]

Fencott, C., Isdale, J., *Design Issues for Virtual Environments*, Proc. of Int. Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D 2001 Conference, Paderborn, 2001.

[Figu02]

Figueroa, P., Green, M., Hoover, H. J., *InTml: A Description Language for VR Applications*, Proc. of 7th Int. Conf. on 3D Web Technology Web3D'2002 (Tempe, 24-28 February 2002), ACM Press, New York, 2002, pp. 53-58.

[Figu04]

Figueroa, P., *Retargeting of Virtual Reality Applications*, Ph.D. Thesis, University of Alberta Alberta, Canada, 2004.

[Figu06]

Figueroa, P., Dachselt, R., Lindt, I., *A Uniform Specification of Mixed Reality Interface Components*, Proc. of the IEEE Virtual Reality Conference VR'2006 (Alexandria, 25-29 March 2006), IEEE Computer Society Press, Los Alamitos, 2006, pp. 289-290.

[Fitz95]

Fitzmaurice, G.W., Ishii, H., Buxton, W., *Bricks: Laying the Foundations for Graspable User Interfaces*, Proc. of ACM Conf. on human Factors in Computing Systems CHI'95 (Denver, 7-11 May 1995), ACM Press, New York, 1995, pp. 442-449.

[Fokk92]

Fokkinga M.M., *A gentle introduction to category theory: the calculational approach*. In Lecture Notes of the STOP 1992 Summer school on Constructive Algorithmic, University of Utrecht, September 1992, pp. 1-72.

[Fole84]

Foley, V.W., Chan, V., *The human factors of computer graphics interaction techniques*, IEEE Computer Graphics & Applications, Vol. 4, 1984, pp. 13-48.

[Fran93]

Frank, M., Foley, J., *Model-based user interface design by example and by answering questions*, Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), ACM Press, New York, 1993, pp 161-162.

References

G

- [Gabb99] Gabbard, J.L., Hix, D., Swan, J.E. II, *User-Centered Design and Evaluation of Virtual Environments*, IEEE Computer Graphics and Applications, Vol. 19, No. 6, Nov. 1999, pp. 51-59.
- [Gaff04] Gaffar, A., Sinnig, D., Seffah, A., Forbrig, P., *Modeling patterns for task models*, Proc. of 3rd Int. Workshop on Task Models and Diagrams for user interface design TAMODIA'2004 (Prague, 15-16 November 2004), Ph. Palanque, P. Slavik, M. Winckler (Eds.), ACM Press, New York, 2004, pp. pp. 99-104.
- [Gamm95] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Booch, G., *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing, Reading, 1995.
- [Garc01] García, F.L.S., Camargo, F., Tissiani, G., *Metodología para criação de ambientes virtuais tridimensionais*, Proc. of 4th Int. Conf. on Graphics Engineering for Arts and Design (Sao Paulo, November 2000).
- [Geig01] Geiger, C., Paelke, V., Reimann, C.C., Rosenbach, W., *Structured Design of Interactive Virtual and Augmented Reality Content*, Proc. of Int. Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D'2001 Conference (Paderborn, 19 February 2001).
- [Geno04] *Genova development Environment*, Genera, Trondheim, 2004. Available on-line at: <http://www.genera.no>.
- [Glas84] Glasersfeld, E.V., *An introduction to radical constructivism*, in "The Invented Reality", P. Watzlawick (Ed.), Norton, New York, 1984, pp. 17-40.
- [Glas91] Glasersfeld, E.V., *Knowing without metaphysics. Aspects of the radical constructivism Position*, in "Research reflexivity", F. Steier (Ed.), SAGE Publishers, London, 1991, pp. 12-29.
- [Glas95] Glasersfeld, E.V., *Radical constructivism: a way of knowing and learning*, Falmer Press, London, 1995.
- [Glas03] Glasersfeld, E.V., *An Exposition of Constructivism: Why Some Like it Radical*, The internet encyclopedia of personal construct psychology of the Scientific Reasoning Research Institute, 2003. Available on-line at: <http://www.oikos.org/constructivism.htm>.
- [Göbe96] Göbel, M., *Industrial Applications of VEs*, IEEE computer graphics and applications, Vol. 16, No. 1, 1996, pp 10-13.
- [Gome03] Gomes de Sousa, L., Leite, J.C., *XICL: a language for the user's interfaces development and its components*, Proc. of the Latin American conference on Human-computer interaction (Rio de Janeiro, 17-20 August 2003), ACM Press, New York, 2003, pp. 191-200.
- [Gome04] Gomes de Sousa, L., Leite, J.C., *XICL- An Extensible Mark-up Language for Developing User Interface and Components*, Proc. of 5th Int. Conf. of Computer-Aided Design of User Interfaces CADUP2004 (Funchal, 13-16 January 2004), Information Systems Series, Kluwer Academics, Dordrecht, 2005, pp. 245-256.

References

- [Gonz06a]
Gonzalez-Calleros, J.M., Vanderdonckt, J., Arteaga, J.M., *A Method for Developing 3D User Interfaces of Information Systems*, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 7, Springer-Verlag, Berlin, 2006, pp. 85-100.
- [Gonz06b]
Gonzalez-Calleros, J.M., *A Method for Developing 3D User Interfaces for Information Systems*, UCL, Louvain-la-Neuve, June 2006.
- [Gonz07a]
Gonzalez-Calleros, J.M., *Model-based Development of Three-Dimensional User Interfaces*, LSM Doctoral Consortium, Mons, 13 December 2007.
- [Gonz08a]
González-Calleros, J.M., Stanciulescu, A., Vanderdonckt, J., Delacré, J.P., Winckler, M., *Comparative Analysis of Transformation Engines for User Interface Development*, Proc. of 4th Int. Workshop on Model-Driven Web Engineering MDWE'2008 (Toulouse, 1 October 2008), N. Koch, G.-J. Houben, A. Vallecillo (Eds.), CEUR Workshop Proceedings, Vol. 389, 2008, pp. 16-30. Available on-line at: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-389/paper02.pdf>
- [Gonz09a]
González-Calleros, J.M., Vanderdonckt, J., Muñoz Arteaga, J., *A structured method for designing usable 3D Web applications*, Chapter 17, in « Usability Engineering for Designing the Web Experience: Methodologies and Principles », T. Spiliotopoulos (Ed.), IGI Global Inc., Hershey, 2009.
- [Gonz09b]
González-Calleros, J.M., Guerrero García, J., Muñoz Arteaga, J., Vanderdonckt, J., Martínez-Ruiz, J., *A Method for Generating Multiplatform User Interfaces for E-Learning Environments*, Chapter 6, in “Multiplatform E-Learning Systems and Technologies: Mobile Devices for Ubiquitous ICT-Based Education“, T.-T. Goh (Ed.), IGI Global Inc., July 2009, pp. 90-111.
- [Gonz09c]
González-Calleros, J.M., Vanderdonckt, J., Muñoz Arteaga, J., *A Structured Approach to Support 3D User Interface Development*, Proc. of 2nd Int. Conf. on Advances in Computer-Human Interactions ACHI'2009 (Cancun, 1-6 February 2009), IEEE Computer Society Press, Los Alamitos, 2009, pp. 75-81.
- [Gonz09d]
González-Calleros, J.M., Vanderdonckt, J., Guerrero-García, J., *Model-based Analysis of Human Errors during Aircraft Cockpit System Design*, LSM Doctoral Consortium, Namur, 12 January 2009.
- [Gonz09e]
González-Calleros, J.M., Guerrero-García, J., Vanderdonckt, J., Muñoz-Arteaga, J., *Towards Canonical Task Types for User Interface Design*, Proc. of Joint 4th Latin American Conference on Human-Computer Interaction-7th Latin American Web Congress LA-Web/CLIHIC'2009 (Merida, 9-11 November 2009), E. Chavez, E. Furtado, A. Moran (Eds.), IEEE Computer Society Press, Los Alamitos, 2009, pp. 63-70.
- [Grac05]
Gracanin, D., Ying, J., *An approach to formal description of user interfaces based on X3D content*, in "Advances in Virtual Environments Technology: Musings on Design, Evaluation, & Applications", K. Stanney, M. Zyda (Eds.), Human Factors/Ergonomics Series, Lawrence Erlbaum Associates, Mahwah, 2005.

References

- [Gram06]
Grammenos, D., Mourouzis, A., Stephanidis, C., *Virtual prints: Augmenting virtual environments with interactive personal marks*, International Journal of Man-Machine Studies, Vol. 64, No. 3, 2006, pp. 221-239.
- [Gree88]
Greenstein, J.S., Arnaut, L.Y., *Input devices*, in "Handbook of Human-Computer Interaction", M. Helander, (Ed.), North-Holland, Amsterdam, 1988, pp. 495-519.
- [Grin96]
Grinstein, G.G., Southard, D.A., *Rapid modelling and design in virtual environments*, Presence, Vol. 5, No. 1, 1996, pp. 146-158.
- [Gros99]
Grosjean, J., Coquillart, S., *The Magic Mirror: A Metaphor for Assisting the Exploration of Virtual Worlds*, Proc. of 15th Spring Conference on Computer Graphics, Comenius University, Bratislava, 1999, pp. 125-129.
- [Grub93]
Gruber, T.R., *A translation Approach to Portable Ontologies*, Knowledge Acquisition, Vol. 5, No. 2, 1993, pp. 199-220.
- [Guer06]
Guerrero-García, J., *Conceptual Modeling of User Information systems of Workflow Information Systems*, DEA in Management Sciences, Université catholique de Louvain, Louvain-la-Neuve, 2006.
- [Guer08a]
Guerrero-García, J., Lemaigre, Ch., Vanderdonckt, J., González-Calleros, J.M., *Towards a Model-Based User Interface Development for Workflow Information Systems*, International Journal of Universal Computer Science, Vol. 14, No. 19, 2008, pp. 3160-3173. Available on-line at: http://www.jucs.org/jucs_14_19/model_driven_approach_to
- [Guer08b]
Guerrero-García, J., Vanderdonckt, J., Gonzalez-Calleros, J.M., *FlowiXML: a Step towards Designing Workflow Management Systems*, Journal of Web Engineering, Vol. 4, No. 2, 2008, pp. 163-182.
- [Guer09a]
Guerrero-García, J., Vanderdonckt, J., González-Calleros, J.M., *Developing user interfaces for community-oriented workflow information systems*, Chapter 16, in D. Akoumianakis (ed.), "Virtual Communities of Practice and Social Interactive Technologies: Lifecycle and Workflow Analysis", IGI Global Inc., Hershey, 2009, pp. 307-329.
- [Guer09b]
Guerrero-García, J., González-Calleros, J.M., Vanderdonckt, J., Muñoz-Arteaga, J., *A Theoretical Survey of User Interface Description Languages: Preliminary Results*, Extended Proc. of 4th Latin American Conference on Human-Computer Interaction CLIHC'2009 (Merida, 9-11 November 2009), A. L. Morán, E. Chávez, E. S. Furtado, M. E. Tentori, M. D. Rodríguez (Eds.), Universidad Autónoma de Baja California, Ensenada, 2009, pp. 8-15.
- [Guil95]
Guillen, M., *Five equations that change the world: the power and Poetry of Mathematics*, Hyperion, New York, 1995.

H

- [Hack98]
Hackos, J. T., Redish, J. C., *User and Task Analysis for Interface Design*, Crystal Dreams Pub, 1998.

References

- [Hard07] Harding-Rolls, P., *Western World MMOG Market: 2006 Review and Forecasts to 2011*, Screen Digest, March 2007. Available on-line at: <http://www.screendigest.com/reports/07westworldmmog/readmore/view.html>
- [Hay] Hay, B., *Creating an Interactive 3D Product Using VRML Tutorial*. Available on-line at: <http://www.virtualrealms.com.au/vrml/tute01/tutorial.htm>
- [Heck02] Heckel, R., Küster, J.M., Taenzer, G., *Confluence of Typed Attributed Graph Transformation Systems*, Proc. of 1st Int. Conf. on Graph Transformations ICGT'2002 (Barcelona, 7-12 October 2002), Lecture Notes in Computer Science, Vol. 2505, Springer-Verlag, Berlin, 2002, pp. 161-176.
- [Heim93] Heim, M., *The Metaphysics of Virtual Reality*, Oxford Press, Oxford, 1993.
- [Helm08] Helms, J., Schaefer, R., Luyten, K., Vanderdonckt, J., Vermeulen, J., Abrams, M., *UIML Version 4.0: Committee Draft*, Available on-line at <http://www.oasis-open.org/committees/download.php/28457/uiml-4.0-cd01.pdf>
- [Herm98] Hermsdorf, D., Gappa, H., Pieper, M., *WebAdapter: A prototype of a WWW-browser with new special needs adaptations*, Proc. of 4th ERCIM Workshop on "User Interfaces for All" UI4All'98 (Stockholm, 19-21 October 1998), ICS-FORTH, 1998, pp. 151-160.
- [Hewe92] Hewett, T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., Verplank, B., *ACM SIGCHI Curricula for Human-Computer Interaction*, Association for Computing Machinery, Special Interest on Computer-Human Interaction 1992.
- [Heyl93] Heylighen, F., *Epistemology, introduction. Principia Cybernetica*, 1993. Available on-line at: <http://pespmc1.vub.ac.be/EPISTEMI.html>.
- [Hirs89] Hirschheim, R., Klein, H.K., *Four Paradigms of Information Systems Development*, Communications of the ACM, Vol. 32, No. 10, October 1989, pp. 1199-1216.
- [Hirs93] Hix, D., Hartson, H.R., *Developing user interfaces: Ensuring usability through product and process*, John Wiley & Sons, New York, 1993.
- [Hoff03] Hoffmann, H., Dachsel, R., Meißner, K., *An Independent Declarative 3D Audio Format on the Basis of XML*, Proc. of the Int. Conf. on Auditory Display ICAD'2003 (Boston, 6-9 July 2003), Boston University Publications Production Department, Boston, 2003, pp. 99-102 Available on-line at: <http://www.icad.org/Proceedings/2003/HoffmannDachsel-2003.pdf>
- [Huma08] HUMAN Project Consortium, *D1.4 Redaction of New Standards Relating to AHMI Designs*. Available on-line at: http://www.human.aero/wp-content/uploads/2009/03/d1_4.pdf
- [Huma09] HUMAN Project Consortium, *D3.4 Virtual Simulation Platform, under preparation*.

References

[Hutc89]

Hutchinson, T.E., White, Jr., K.P., Martin, W.N., Reichert, K.N., Frey, L.A., *Human-Computer Interaction Using Eye-Gaze Input*, IEEE Transactions on systems, man, and cybernetics, Vol. 19, No. 6, 1989, pp. 1527-1533.

I

[IBM02]

IBM (2002), WSXL specification, April 2002, retrieved on January 2nd 2009.

[Iran00]

Irani, P., Ware, C. *Diagrams based on structural object perception*, Proc. of the ACM Working Conf. on Advanced Visual Interfaces AVI'2000 (Palermo, 23-26 May 2000), ACM Press, New York, 2000, pp. 61-67.

J

[Jans06]

Jansen, B. J., *Using Temporal Patterns of Interaction to Design Effective Automated Searching engines*, Communications of the ACM, Vol. 49, No. 4, April 2006, pp. 72-74.

[Jean06]

Jeandrain, A.C., *L'influence de la présence subjective du consommateur sur sa perception des points de vente virtuels : Une application au commerce de détail*, PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, 2006.

[John95]

Johnsgard, T.J., Page, S., R., Wilson, R.D., Zeno, R. J., *A Comparison of Graphical User Interface Widgets for Various Tasks*, Proc. of the 39th Annual Meeting of the Human Factors & Ergonomics Society HFES'95 (Santa Monica, 1995), Human Factors and Ergonomics Society, Santa Monica, 1995, pp. 287-291.

[John92]

Johnson, P., Markopoulos, P., Johnson, H., *Task knowledge structures: A specification of user task models and interaction dialogues*, Proc. of 11th Int. Workshop on Informatics and Psychology, Task Analysis in Human-Computer Interaction (Schraeding, June 9-11), 1992.

[John01]

Johnson, M., Dampney, O., *Category theory as a (meta) ontology for information systems*, Proc. of the Int. Conf. on Formal Ontology in Information Systems FOIS'01 (Ogunquit, 2001), pp. 59-69.

[Jorg99]

Jorgensen, Ch., Wheeler, K., Stepniewski, S., *Bioelectric Control of a 757 Class High Fidelity Aircraft Simulation*. Retrieved from the web at: <http://ic.arc.nasa.gov/publications/index.html>, 1999.

K

[Kahn92]

Kahneman, D., Triesman, A., Gibbs, B.J., *The reviewing of objects files: Objects-specific integration of information*, Cognition Psychology, Vol. 24, 1992, pp. 175-219.

References

- [Kakl08a] Kaklanis, N., González-Calleros, J.M., Vanderdonckt, J., Tzovaras, D., *Hapgets, Towards Haptically-enhanced widgets Based on a User Interface Description Language*, Proc. of Workshop on Multimodal Interaction Through Haptic Feedback MITH'2008 (Naples, 31 May 2008).
- [Kakl08b] Kaklanis, N., González-Calleros, J.M., Vanderdonckt, J., Tzovaras, D., *Haptic Rendering Engine of Web Pages for Blind Users*, Proc. of 9th Int. Conf. on Advanced Visual Interfaces AVI'2008 (Naples, 28-30 May 2008), ACM Press, New York, 2008, pp. 437-440.
- [Kats03] Katsurada, K., Nakamura, Y., Yamada, H., Nitta, T., *XISL: A Language for Describing Multimodal Interaction Scenarios*, Proc. of 5th Int. Conf. on Multimodal Interfaces ICMI'2003 (Vancouver, 5-7 November 2003), ACM Press, New York, 2003, pp. 281–284.
- [Kaur97] Kaur, K., *Designing Virtual Environments for Usability*, Proc. of IFIP TC13 Int. Conf. on Human-Computer Interaction Interact'97 (Sydney, 14-18 July 1997), Chapman & Hall London, 1997, pp. 636-639.
- [Kats98] Kaur, K., *Designing virtual environments for usability*, Ph. D. Thesis, City University, London, 1998.
- [Kaur99] Kaur, K., Maiden, N.A.M., Sutcliffe, A.G., *Interacting with virtual environments: an evaluation of a model of interaction*, Interacting with Computers, Vol. 11, No. 4, 1999, pp. 403-426.
- [Kels03] Kelso, J., Satterfield, S., Arsenault, L., Ketchan, P., Kriz, R., *DIVERSE: A framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations*, Presence: Teleoper. Virtual Environ., Vol. 12, No. 1, 2003, pp. 19–36.
- [Kenn93] Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G., *Simulator sickness questionnaire (SSQ): A new method for quantifying simulator sickness*, International Journal of Aviation Psychology, Vol. 3, 1993, pp. 203-220.
- [Khaz00] Khazanchi, D., Munkvold, B.E., *Is Information Systems a Science? An Inquiry into the Nature of the Information Systems Discipline*, The DATA 24 BASE for Advances in Information Systems, Vol. 31, No. 3, 2000.
- [Kier97] Kieras, D.E., Meyer, D.E., *An overview of the EDIC architecture for cognition and performance with application to human-computer interaction*, Human-Computer Interaction, Vol. 12, 1997, pp. 391-438.
- [Kim98] Kim, G.J., Kang, K.C., Kim, H., Lee, J., *Software Engineering of Virtual Worlds*, Proc. of ACM Symposium on Virtual Reality Software and Technology VRST'98 (Taipei, November 1998), ACM Press, New York, 1998, pp. 131-139.
- [Kiyo00] Kiyokawa, K., Takemura, H., Yokoya, N., *Seamless Design for 3D object creation*, IEEE multimedia, Vol. 7, No. 1, January-March 2000, pp. 22-33.

References

- [Krei01] Kreitler, S., *Psychological Perspective on Virtual Reality*, in "Virtual Reality: Cognitive Foundations, Technological Issues & Philosophical Implications", Riegler, A., Peschl, M., Edlinger, K., Fleck, G., Feigl, W. (Eds.), Peter Lang, Frankfurt am Main, 2001.
- [Krui00] Kruijff, E., *Wayfinding*, in SIGGRAPH 2000 Course 3D User Interface Design: Fundamental Techniques, Theory, and Practice, Doug Bowman, Ernst Kruijff, Joseph LaViola, Ivan Poupyrev, and Mark Mine, 2000.
- L**
- [Laha08] Lahav, O., Mioduser, D., *Haptic-feedback support for cognitive mapping of unknown spaces by people who are blind*, Int. J. Human-Computer Studies, Vol. 66, 2008, pp. 23–35.
- [Lari03] Larimer, D., Bowman, D., *VEWL: A Framework for Building a Windowing Interface in a Virtual Environment*, Proc. of IFIP TC13 Int. Conf. on Human-Computer Interaction Interact'2003 (Zürich, 1-5 September 2003), IOS Press, Amsterdam, 2003, pp. 809–812.
- [Lato08] Latoschik, M.E., Reiners, D. Blach, R., Figueroa, P., Dachselt, R., *Workshop SEARIS - Software Engineering and Architectures for Realtime Interactive Systems*, IEEE Virtual Reality Conf. VR'2008, 2008.
- [Laud09] Laudon, K.C., Laudon, J.P., *Management Information Systems: Managing the Digital Firm*, Pearson, 2009.
- [Lavi08] LaViola, J., *Bringing VR and Spatial 3D Interaction to the Masses through Video Games*, IEEE Computer Graphics and Applications, Vol. 28, No. 5, Sept./Oct. 2008, pp. 10-15.
- [Lecu03] Lecuyer, A., Mobuchon, P., Megard, C., Perret, J., Andriot, C., Colinot, J.-P., *HOMERE: a multimodal system for visually impaired people to explore virtual environments*, Proc. of IEEE Int. Conf. on Virtual Reality VR'2003 (Washington, 22-26 March 2003), IEEE Computer Society Press, Los Alamitos, 2003, pp. 251–258.
- [Lema08] Lemaigre, Ch., Guerrero, J., Vanderdonckt, J., *Interface Model Elicitation from Textual Scenarios*, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 53-66.
- [Leno84] Lenorovitz, D.R., Phillips, M.D., Ardrey, R.S., Kloster, G.V., *A taxonomic approach to characterizing human-computer interaction*, in "Human-Computer Interaction", G. Salvendy (Ed.), Elsevier Science Publishers, Amsterdam, 1984, pp.111-116.
- [Lewi95] Lewis, J.R., *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use*, International Journal of Human-Computer Interaction, Vol. 7, No. 1, 1995, pp. 57-78.

References

- [Limb04a] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V., *USIXML: a Language Supporting Multi-Path Development of User Interfaces*, Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, 11-13 July 2004), Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200-220.
- [Limb04b] Limbourg, Q. & Vanderdonckt, J., *UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence*, in "Engineering Advanced Web Applications", Matera, M., Comai, S. (Eds.), Rinton Press, Paramus, 2004, pp. 325-338. Available on-line at: <http://www.isys.ucl.ac.be/bchi/publications/2004/Limbourg-JWE2004.pdf>
- [Limb04c] Limbourg, Q., *Multi-Path Development of User Interfaces*, PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, 2004. Available on-line at: <http://www.isys.ucl.ac.be/bchi/publications/Ph.D.Theses/Limbourg-PhD2004.pdf>
- [Limb04d] Limbourg, Q., Vanderdonckt, J., *Addressing the Mapping Problem in User Interface Design with UsiXML*, Proc. of 3rd Int. Workshop on Task Models and Diagrams for user interface design TAMODIA'2004 (Prague, 15-16 November 2004), Ph. Palanque, P. Slavik, M. Winckler (Eds.), ACM Press, New York, 2004, pp. 155-163.
- [Limb04e] Limbourg, Q., Vanderdonckt, J., *Transformational Development of User Interfaces with Graph Transformations*, Proc. of 5th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2004 (Funchal, 14-16 January 2004), Kluwer Academics Pub., Dordrecht, 2005, pp. 105-118.
- [Luo95] Luo, P., *A Human-Computer Collaboration Paradigm for Bridging Design Conceptualization and Implementation*, in "Interactive Systems: Design, Specification, and Verification", Proc. of the 1st Eurographics Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'94 (Bocca di Magra, 8-10 June 1994), Paternò F. (Ed.), Springer-Verlag, Berlin, pp. 129-147.
- [Luyt03] Luyten, K., Clerckx, T., Coninx, K., Vanderdonckt, J., *Derivation of a Dialog Model from a Task Model by Activity Chain Extraction*, Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4-6 June 2003), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 203-217.
- [Luyt04a] Luyten, K., Abrams, M., Limbourg, Q., Vanderdonckt, J. (Eds.), Proceedings of the ACM AVI'2004 Workshop Developing User Interfaces with XML: Advances on User Interface Description Languages UIXML'04 (Gallipoli, 25 May 2004), Gallipoli, 2004.

M

- [Mack91] Mackinlay, J.D., Robertson, G.G., Card, S.K., *The Perspective Wall: Detail and Context Smoothly Integrated*, Proc. of the ACM Conference on Human factors in computing systems CHI'91 (New Orleans, 1991), ACM Press, New York, 1991, pp. 173-176.

References

- [Mac191] MacLean, A., Young, R.M., Bellotti, V., Moran, T.P., *Questions, Options, and Criteria: Elements of Design Space Analysis*, Human-Computer Interaction, Vol. 6, No. 3-4, 1991, pp. 201–250.
- [Magn06] Magnusson, C., Danielsson, H., Rassmus-Gröhn, K., *Non Visual Haptic Audio Tools for Virtual Environments*, Proc. of 1st Workshop on Haptic and Audio Interaction Design HAID'2006 (Glasgow, 31 August - 1 September 2006), Lecture Notes in Computer Science, Vol. 4129, Springer, Berlin, 2006, pp. 111–120.
- [Mahe01] Mahemoff, M.J., Johnston, L.J., *Usability Pattern Languages: the "Language" Aspect*, Proc. of 8th TC13 IFIP Int. Conf. on Human-Computer Interaction Interact'2001 (Tokyo, 9-13 July 2001), IOS Press, Amsterdam, 2001, pp. 350-358. Available on-line at: <http://mahemoff.com/paper/language/language.pdf>
- [Mao00] Mao, X., Hatanaka, Y., Imamiya, A., Kato, Y., Go, K., *Visualizing Computational Wear with Physical Wear*, Proc. of 6th ERCIM Workshop on "User Interfaces for All" UI4All'00 (Florence, October 25-26, 2000), P. L. Emiliani, C. Stephanidis (Eds.), ERCIM, 2000. Available on-line at: http://ui4all.ics.forth.gr/UI4ALL-2000/files/Long_papers/Mao.pdf
- [Mape95] Mapes, D., Moshell, J. *A Two-Handed Interface for Object Manipulation in virtual Environments*, Presence: Teleoperators and Virtual Environments, Vol. 4, No. 4, 1995, pp. 403-426.
- [Maqu04] Maquil, V., *Automatic Generation of Graphical User Interfaces in Studierstube*, BSc thesis, Interactive Media Systems Group, Institute for Software Technology and Interactive Systems, Vienna University of Technology, 2004. Available on-line at: <https://www.ims.tuwien.ac.at/media/documents/publications/MaquilBachelorThesis.pdf>.
- [Marq97] Marquis J.-P., *Stanford encyclopedia of philosophy: Category theory*, 1997. Available on-line at: <http://plato.stanford.edu/entries/category-theory/>.
- [Mey97] Meyer, B., *Object-Oriented Software Construction*, Prentice Hall, Upper Saddle River, 1997.
- [Mens99] Mens, T., *A Formal Foundation for Object-Oriented Software Evolution*, PhD thesis, Vrije Universiteit Brussel, Brussels, 1999.
- [Mey85] Meyer, B., *On Formalism in Specification*, IEEE Software, Vol. 2, No. 1, 1985, pp. 6-25.
- [Mile79] Miles, M.B., *Qualitative Data as an Attractive Nuisance: The Problem of Analysis*, Administrative Science Quarterly, Vol. 24, No. 4, 1979, pp. 590-601.
- [Milg94] Milgram, P., Kishino, F., *A Taxonomy of Mixed-Reality Visual Displays*, IEICE Transactions on Informations Systems, E77-D(12), 1994. Available on-line at: http://vered.rose.utoronto.ca/people/paul_dir/IEICE94/ieice.html.
- [Mill98] Miller, T., Zeleznik, R.C., *An Insidious Haptic Invasion: Adding Force Feedback to the X Desktop*, Proc. of ACM Conf. on User Interface Software and Technology UIST'98 (San Francisco, Nov. 1998), ACM Press, New York, 1998, pp. 59–64.

References

- [Mill03] Miller, J., Mukerij J., *MDA Guide version 1.0.1*, 2003. Available on-line from at: www.omg.org.
- [Mine95] Mine, M., *Virtual environment interaction techniques*, UNC Chapel Hill CS Dept., Technical Report TR95-018, 1995.
- [Moli02] Molina, P.J., Meliá, S., Pastor, O., *Just-UI: A User Interface Specification Model*, in "Computer-Aided Design of User Interfaces III", Proc. of 4th Int. Conf. of Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002), Kolski, Ch., Vanderdonckt, J. (Eds.), Information Systems Series, Kluwer Academics, Dordrecht, 2002, pp. 63-74.
- [Moli05] Molina, J.P., Vanderdonckt, J., Montero, F., Gonzalez, P., *Towards Virtualization of User Interfaces based on UsiXML*, Proc. of 10th ACM Int. Conf. on 3D Web Technology Web3D'2005 (Bangor, 29 March-1 April 2005), ACM Press, New York, 2005, pp. 169-178.
- [Moli06] Molina, J.P., Vanderdonckt, J., González, P., Fernández Caballero, A., Lozano Pérez, D., *Rapid Prototyping of Distributed User Interfaces*, Chapter 12, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Information Systems Series, Springer-Verlag, Berlin, 2006, pp. 85-100.
- [Moli08] Molina, J.P., *A Structured Approach to the Development of 3D User Interfaces*, Ph.D. thesis, University of Castilla-La Mancha, Albacete, Spain, 29 February 2008.
- [Mont70] Montanari, U.G., *Separable Graphs, planar Graphs and Web Grammars*, Inf. Contr., Vol. 16, 1970, pp. 243-267.
- [Mont05] Montero, F., López-Jaquero, V., Vanderdonckt, J., Gonzalez, P., Lozano, M.D., Limbourg, Q., *Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML*, Proc. of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005 (Newcastle upon Tyne, 13-15 July 2005), S.W. Gilroy, M.D. Harrison (Eds.), Lecture Notes in Computer Science, Vol. 3941, Springer-Verlag, Berlin, 2005, pp. 161-172.
- [More07] Moreno, N., Romero, J.R., Vallecillo, A., *An overview of Model Driven Web Engineering and the MDA*, in "Web Engineering and Web Applications Design Methods", L. Olsina, O. Pastor, G. Rossi, D. Schwabe (Eds.), Volume 12, Human-Computer Interaction Series, Chapter 12, Springer, Berlin, 2007.
- [Mori04] Mori, G., Paternò, F., Santoro, C., *Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions*, IEEE Transactions on Software Engineering, Vol. 30, No. 8, August 2004, pp. 1-14.
- [Mull98] Mullay, J., *IBM RealThings*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'98 (Los Angeles, 1998), ACM Press, New York, 1998, pp. 13-14.
- [Murp97] Murphy, E., *Constructivism from Philosophy to Practice*, 1997. Available on-line at: <http://www.cdli.ca/~elmurphy/emurphy/cle.html>.

References

- [Myer00] Myers, B., Hudson, S. E., Pausch, R., *Past, Present and Future of User Interface Software Tools*, ACM Transactions on Computer Human Interaction, Vol. 7, No. 1, March, 2000, pp. 3-28.
- [Myer92] Myers, B., Rosson, M., *Survey on User Interface Programming*, Proc. of ACM Conference on Human Factors in Computing Systems CHI'92 (Monterey, May 1992), ACM Press, New York, 1992, pp. 195-202.
- [Myna94] Mynatt, E.D., *Auditory presentations of graphical user interfaces*, in "Auditory Display: Sonification, Audification, and Auditory Interfaces", G. Kramer (Ed.), Addison-Wesley, Reading, 1994, pp. 533-556.
- N**
- [Nade97] Nadeau, D.R., *Introduction to VRML 97*, Eurographics 97, tutorial notes sections, 1997.
- [Neal01] Neale, H., Nichols, S., *Designing and developing Virtual Environments: methods and applications*, Proc. of Visualization and Virtual Environments Community Club (VVECC) Workshop, Designing of Virtual Environments, 2001.
- [Nede06] Nedel, L. P., Freitas, C.M.D.S., *3D User Interfaces: from Pragmatics to Formal Description*. Research in Interactive Design, Springer-Verlag, Vol. 1, 2006, pp. 1-13.
- [Nesb01] Nesbitt, K.V., *Modeling the multi-sensory design space*, Proc. of the 2001 Asia-Pacific Symposium on Information Visualisation InVis.au'2001 (Sydney, 3-4 December 2001), ACM International Conference Proceeding Series, Vol. 16, Australian Computer Society, Darlinghurst, 2001, pp. 27-36.
- [Nexo09] NEXOF-RA Investigation Team on Declarative UI Authoring and Context Models. *Set of Reference Models for Interactive Applications*, 2009. Available on-line at: http://forge.morfeo-project.org/wiki/en/index.php/Interactive_Application_Models
- [Niga95] Nigay, L., Coutaz J., *A Generic Platform for Addressing the Multimodal Challenge*, Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'95 (Denver, 7-11 May 1995), ACM Press, New York, 1995, pp. 98-105.
- [Norm86] Norman, D.A., *Cognitive Engineering*, in "User Centered System Design", D.A. Norman, S.D. Draper (Eds.), Lawrence Erlbaum Associates, 1986, pp. 31-61.
- O**
- [Ogre05] The OGRE team, *About OGRE*, 2005. Available on-line at: http://www.ogre3d.org/index.php?option=com_content&task=view&id=19&Itemid=79

References

- [OMod97]
O'Modhrain, M.S., Gillespie, R.B., *The Moose: A Haptic User Interface for Blind Persons*, Proc. of the 3rd WWW6 conference (Santa Clara, 1997). Available on-line at: <http://ccrma.stanford.edu/STANM/stanms/stanm95/stanm95.pdf>
- [Open04]
OpenGL, The industry standard for 2D and 3D Graphics, *OpenGL Overview*. 2004. Available on-line at: <http://www.opengl.org/about/overview.html>
- [Oula09]
Oulasvirta, A., Nurminen, A., Nivala, A.M., *Embodied interaction with a 3D versus 2D mobile map*, Personal and Ubiquitous Computing, Vol. 13, No. 4, May 2009, pp. 303 - 320.
- P**
- [Pate97]
Paternò F., Mancini C., Meniconi S., *ConcurTaskTree: A diagrammatic notation for specifying task models*, Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'97 (Sydney, 14-18 July 1997), Kluwer Academic Publishers, Boston, 1997, pp. 362-369.
- [Pate99]
Paternò, F., *Model-based design and evaluation of interactive applications*, Applied Computing, Springer, Berlin, 1999.
- [Pate03]
Paternò, F., Santoro, C., *A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms*, Interacting with Computers, Vol. 15, No. 4, 2003, pp. 349-366.
- [Paus95]
Pausch, R., Burnette, T., Brockway, D., Weiblen, M.E., *Navigation and locomotion in virtual worlds via flight into hand-held miniatures*, Proc. of the 22nd Annual Conf. on Computer graphics and interactive techniques GIIT'95, ACM Press, new York, 1995, pp. 399-400.
- [Park98]
Parker, G., Franck, G., Ware, C., *Visualization of Large Nested Graphs in 3D: Navigation and Interaction*, J. Visual Languages and Computing, Vol. 9, No. 3, 1998, pp. 299-317.
- [Parr07]
Parris, C., *Do better business in 3D*, Business Week, 1 May 2007. Available on-line at: http://www.businessweek.com/technology/content/may2007/tc20070501_526224.htm
- [Pell05a]
Pellens, B., De Troyer, O., Bille, W., Kleinermann, F., Romero, R., *An Ontology-Driven Approach for Modeling Behavior in Virtual Environments*, OTM Workshops, 2005, pp. 1215-1224
- [Pell05b]
Pellens, B., Bille, W., De Troyer, O., Kleinermann, F.: *VR-WISE: A Conceptual Modelling Approach For Virtual Environments*, CD-ROM Proceedings of the Methods and Tools for Virtual Reality Workshop MeTo-VR'2005 (Gent, 2005).
- [Pere00]
Pereira, A.T.C. Rebelo, I.B., Tissiani, G., *Design de Interfaces para Ambientes Virtuais: Como obter Usabilidade em 3D*, Actas de la IV Sociedad Iberoamericana de Gráfica Digital SI-GraDI'2000 (Rio de Janeiro, Septiembre 2000), pp. 25-28.

References

- [Pesc01a] Peschl, M.F., Riegler A., *Virtual Science. Virtuality and knowledge acquisition in science and cognition*, in "Virtual Reality: Cognitive Foundations, Technological Issues & Philosophical Implications", Riegler, A., Peschl, M., Edlinger, K., Fleck, G., Feigl, W. (Eds.), Peter Lang, Frankfurt am Main, 2001.
- [Pesc01b] Peschl, M.F., *Constructivism, Cognition, and Science-An Investigation of its Links and Possible Shortcomings*, in Foundations of Science, Riegler (ed.), Special issue on The impact of Radical Constructivism on Science, Vol. 6, No. 1-3, 2001, pp. 125-161.
- [Pica03] Picard, E., Fierstone, J., Pinna-Dery, A-M., Riveill, M., Atelier de composition d'IHM et évaluation du modèle de composants, Livrable I3, Project RNTL ASPECT, Laboratoire I3S, May 2003.
- [Pier97] Pierce, J. S., Forsberg, A. S., Conway, M. J., *Image Plane Interaction Techniques in 3D Immersive Environments*, Proc. of the Symposium on Interactive 3D graphics, 1997.
- [Poly05] Polys, N.F., Hetherington, R., Brutzman, D., Gracacin, D., *Engineering Virtual Environments with X3D*, Tutorial at ACM Web3D 2005 Symposium, 10th Int. Conf. on 3D Web Technology, Bangor, March 2005. Available on-line at: http://people.cs.vt.edu/~bowman/3dui.org/web3d2005/X3D_IT_Tutorial/
- [Poup96] Poupyrev, I., Billinghamurst, M., Weghorst, S., Ichikawa, T., *The Go-Go Interaction Technique: NonLinear Mapping for Direct Manipulation in VR*, Proc. of ACM Symposium on User Interface Software Technology UIST'96, ACM Press, New York, 1996, pp. 79-80.
- [Poup97] Poupyrev, I., Weghorst, S., Billinghamurst, M., Ichikawa, T., *A framework and testbed for studying manipulation techniques for immersive VR*, Proc. of the ACM Symposium on Virtual reality software and technology VRST'97, ACM Press, New York, 1997, pp. 21-28.
- [Poup98] Poupyrev, I., Weghorst, S., Billinghamurst, M., Ichikawa, T., *Egocentric Object Manipulation in Virtual Environments : Empirical evaluation of Interaction Techniques*, Proc. of EUROGRAPHICS' 98, Comput. Graph. Forum, Vol. 17, No. 3, 1998, pp. 41-52.
- [Poup00] Poupyrev, I. *3D Manipulation Techniques*, in 3D UI Course Notes, Doug Bowman, Ernst Kruijff, Joseph LaViola, Ivan Poupyrev, and Mark Mine, 2000.
- [Poup01] Poupyrev, I. *3D Manipulation Techniques*, in 3D UI Course Notes, Doug Bowman, Ernst Kruijff, Joseph LaViola, Ivan Poupyrev, and Mark Mine, 2001.
- [Prib03] Pribeanu, C., Vanderdonckt, J., *A Pattern-based Approach to User Interface Development*, Proc. of 2nd Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2003, included in Proc. of 2nd Int. Conf. on Universal Access in Human-Computer Interaction UAHCI'2003 (Crete, 22-27 June 2003), Vol. 4, C. Stephanidis (ed.), Lawrence Erlbaum Associates, Mahwah, 2003, pp. 1524-1528..
- [Puer96] Puerta, A.R., *The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development*, Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996), Presses Universitaires de Namur, 1996, pp. 19-35.

References

- [Puer97]
Puerta, A.R., *A Model-Based Interface Development Environment*, IEEE Software, Vol. 14, No. 4, 1997, pp. 41–47. Available on-line at : <http://www.arpuerta.com/pubs/ieee97.htm>
- [Puer02]
Puerta, A., Eisenstein, J., *XIML: A common representation for interaction data*, Proc. of the 7th Int. Conf. on Intelligent User Interfaces IUI'2002, ACM Press, New York, 2002, pp. 69–76.
- R**
- [Rada07]
Radake, F., Forbrig, P., Seffah, A., Sinning, D., *PIM tool: Support for pattern-Driven and Model-Based UI Development*, Proc. of TaMoDia'2006, Lecture Notes in Computer Science, Vol. 4385, Springer, Berlin, 2007, pp. 82–96.
- [Rams96a]
Ramstein, C., Century, M., *Navigation on the Web using Haptic Feedback*, Proc. of the Int. Symposium on Electronic Art ISEA'96.
- [Rams96b]
Ramstein, C., Martial, O., Dufresne, A., Carignan, M., Chassé, P., Mabilieu, P., *Touching and hearing GUIs - Design Issues*, Proc. of the ACM Int. Conf. on Assistive Technologies ASSETS'96, ACM Press, New York, 1996, pp. 2–9.
- [Raut00]
Rauterberg, G.W.M., *How to characterize a research line for user-system interaction*, IPO Annual Progress Report 35, 2000.
- [Redd05]
Reddy, M., *3D Graphics and the Film Production Pipeline*. Invited talk, ACM Web3D 2005 Symposium, 10th Int. Conf. on 3D Web Technology, Bangor, March-April 2005.
- [Reis06]
Reiss, M., Moal, M., Barnard, Y., Ramu, J.-Ph., Froger, A., *Using Ontologies to Conceptualize the Aeronautical Domain*, Proc. of the Int. Conf. on Human-Computer Interaction in Aeronautics HCIAero'2006, Cépaduès-Éditions, Toulouse, 2006, pp. 56-63.
- [Reki95]
Rekimoto, J., Nagao, K., *The World through the Computer: Computer Augmented Interaction with Real World Environments*, Proc. of ACM Symposium on User Interface Software Technology UIST'95, ACM Press, New York, 1995, pp. 29-36.
- [Reki99]
Rekimoto, J., Saitoh, M., *Augmented surfaces: A spatially continuous work space for hybrid computing environments*, Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'99, ACM Press, New York, 1999, pp. 378-385.
- [Robe83]
Roberts, N., Andersen, D., Deal, R., Garet, M., Shaffer, W., *Introduction to Computer Simulation, a system dynamics modelling approach*, Productivity Press, 1983.
- [Ross02]
Rosson, M.B., Carroll, J., *Usability Engineering: Scenario Based Development of Human-Computer Interaction*, Morgan Kaufmann, San Francisco, 2002.

References

S

- [Sall06] Sallnäs, E., Bjerstedt-Blom, K., Winberg, F., Severinson-Eklundh, K., *Navigation and Control in Haptic Applications Shared by Blind and Sighted Users*, Proc. of 1st Workshop on Haptic and Audio Interaction Design HAID'2006 (Glasgow, 31 August-1 September 2006), Lecture Notes in Computer Science, Vol. 4129, Springer, Berlin, 2006, pp. 68–80.
- [Salm08] Salminen, K., Surakka, V., Lylykangas, J., Raisamo, J., Saarinen, R., Raisamo, R., Rantala, J., Evreinov, G., *Emotional and Behavioral Responses to Haptic Stimulation*, Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'2008 (Florence, 5-10 April 2008), ACM Press, New York, 2008, pp. 1555–1562.
- [Schae06] Schaefer, R., Steffen, B., Wolfgang, M., *Task Models and Diagrams for User Interface Design*, Proc. of 5th International Workshop TAMODIA'2006 (Hasselt, October 2006), Lecture Notes in Computer Science, Vol. 4385, Springer Verlag, Berlin, 2006, pp. 39-53.
- [Schl96] Schlungbaum, E., *Model-based user interface software tools - current state of declarative models*, Technical Report GIT-GVU 96-30, Georgia Institute of Technology, 1996.
- [Schu01] Schuemie, M. J., van der Straaten, P., Krijn, M., van der Mast, C.P.G., *Research on Presence in VR: A Survey*, Cyberpsychology and Behavior, Vol. 4, No. 2, 2001, pp. 183-202.
- [Shne98] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson Education, 1998.
- [Shne02] Shneiderman, B., *3D or Not 3D: When and Why Does it Work?*. Human-Computer Interaction Laboratory & Department of Computer Science University of Maryland. Talk in Web3D (Phoenix, 26 February 2002).
- [Shne03] Shneiderman, B., *Why Not Make Interfaces Better than 3D Reality*, Virtualization Viewpoints. Theresa-Marie Rhyme (Ed.), November-December, 2003.
- [Shne04] Shneiderman, B., Plaisant, C., *Designing the User Interface*. 4th Edition, Addison Wesley, Reading, 2004.
- [Shne06] Shneiderman, B., Fischer, G., *Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop*, International Journal of Human-Computer Interaction, Vol. 20, No. 2, 2006, pp. 61–77.
- [Sili03] Silicon Graphics, *Open Inventor web site*, 2003. Available on-line at : <http://oss.sgi.com/projects/inventor/>
- [Sjö02] Sjöström, C., *Non-Visual Haptic Interaction Design - Guidelines and Applications*, Ph.D Thesis, Lund Institute of Technology, Sweden, 2002. Available on-line at <http://www.certec.lth.se/doc/hapticinteraction/>

References

- [Slat99] Slater, M., *Measuring Presence: A Response to the Witmer and Singer Presence Questionnaire*, Presence: Teleoperators and Virtual Environments, Vol. 8, No.5, 1999, pp. 560-565.
- [Smi06] Smith, S. P., Willans, J.S., *Virtual Objects Specification for Usable Virtual Environments*, Proc. of OZCHI'2006 (Sydney, 2006).
- [Smi04] Smith, S. P., Marsh T., *Evaluating design guidelines for reducing user disorientation in a desktop virtual environment*, Virtual Reality, Vol. 8, No. 1, 2004, pp. 55-62.
- [Smi01] Smith, S., Duke, D., *Design Support for Virtual Environments*. Visualization and Virtual Environments Community Club (VVECC) Workshop: Design of Virtual Environments, Oxfordshire, 2001.
- [Somm99] Sommerville, I., *Software Engineering*, 5th edition, Addison Wesley, Reading, 1999.
- [Sott07] Sottet, J.-S., Calvary, G., Coutaz, J., Favre, J.-M., Vanderdonckt, J., Stanculescu, A., Lepreux, S. *A Language Perspective on the Development of Plastic Multimodal User Interfaces*, Journal of Multimodal User Interfaces, Vol. 1, No. 2, 2007, pp. 1-12.
- [Souc03] Souchon, N., Vanderdonckt, J., *A Review of XML-Compliant User Interface Description Languages*, Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4-6 June 2003), J. Jorge, N.J. Nunes, J. Cunha (Eds.), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 377-391.
- [Sowa92] Sowa, J. F., *Conceptual Graphs Summary*, in "Conceptual Structures: Current Research and Practice", Eklund P., Nagle T., Nagle J., and Gerholz L. (Eds.), Ellis Horwood, 1992, pp. 3-52.
- [Spai91] Spain E., Holzhausen, K., *Stereoscopic versus orthogonal view displays for performance of a remote manipulation task*, Proc. of Stereoscopic Displays and Applications II, SPIE, 1991, pp. 103-110.
- [Sphe05] Sphere site, Available on-line at : <http://www.spheresite.com/>
- [Stan06] Stanculescu, A., *A Transformational approach for developing multimodal web interfaces*, DEA in Management Sciences Thesis, Université catholique de Louvain, Louvain-la-Neuve, 2006.
- [Stan08] Stanculescu, A., *A Methodology for Developing Multimodal User Interfaces of Information Systems*, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, 25 June 2008.
- [Stee98] Steed, A., Tromp, J., *Experiences with the evaluation of CVE applications*, Proc. of the Conf. Collaborative Virtual Environments CVE'98 (Manchester, 17-19 June 1998).
- [Stee08] Steed, A., *Some useful abstractions for re-usable virtual environment platforms*, Proc. of the IEEE Virtual Reality Workshop for Software Engineering and Architectures for Realtime Interactive Systems, 2008.

References

- [Stoa95] Stoakley, R., Conway, M.J., Pausch, R., *Virtual Reality on a WIM: Interactive Worlds in Miniature*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'95 (Denver, 7-11 May 1995), ACM Press, New York, 1995, pp. 265-272.
- [Stre86] Streibel, M. J., *A critical analysis of the use of computers in education*, Educational Communications and Technology Journal, Vol. 34, No. 3, 1986.
- [Stud] Studierstube Augmented Reality Project, Available on-line at: <http://www.studierstube.org/>.
- [Sutc03] Sutcliffe, A., *Multimedia and Virtual Reality: Designing Multisensory User Interfaces*, Lawrence Erlbaum Associates, Mahwah, 2003.
- [Sutc96] Sutcliffe, A., Patel, U., *3D or Not 3D: Is It Nobler in the Mind?*, Proc. British Human-Computer Interaction HCI'96, Cambridge Univ. Press, Cambridge, 1996, pp. 79-94.
- [Sutc94] Sutcliffe, A., Faraday, P., *Designing Presentation in Multimedia Interfaces*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'94 (Boston, April 1994), ACM Press, New York, 1994, pp. 92-98.
- [Sun05] Sun Micro systems, *The Java™ tutorial: Listeners Supported by Swing Components*. Available on-line at: http://java.sun.com/docs/books/tutorial/uiswing/events/eventsand_components.html
- [Swan05] Swan, J.E., Gabbard, J.L. *Survey of User-Based Experimentation in Augmented Reality?* Proc. of 1st Int. Conf. on Virtual Reality (Las Vegas, 2005).
- [Szek96] Szekeley, P., *Retrospective and challenges for model-based interface development*, Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996), J. Vanderdonck (Ed.), Presses Universitaires de Namur, Namur, 1996.
- T**
- [Tan01] Tan, D.S., Robertson, G.G., Czerwinski, M., *Exploring 3D navigation: Combining speed-coupled flying with orbiting*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2001 (Seattle, March 2001), ACM Press, new York, 2001.
- [Teor86] Teory, T.J., Yang, D., Fry, J.P. *A Logical Design Methodology for Relational Databases using the Extended Entity-Relationship Model*, ACM Computing Surveys, Vol. 18, No. 2, June 1986, pp. 197-222.
- [Thev01] Thevenin, D., *Adaptation en Interaction Homme-Machine: le cas de la Plasticité*, Ph.D. thesis, Université Joseph Fourier, Grenoble, 2001. Available on-line at: : <http://iuhm.imag.fr/publs/2001>.
- [Thié03] Thiéart, R.A. et al., *Méthodes de Recherche en Management*, Dunod, Paris. 2003.

References

[Tikk06]

Tikka, V., Laitinen, P., *Designing Haptic Feedback for Touch Display: Experimental Study of Perceived Intensity and Integration of Haptic and Audio*. Proc. of 1st Int. Workshop on Haptic and Audio Interaction Design HAID'2006 (Glasgow, 31 August-1 September 2006), Lecture Notes in Computer Science, Vol. 4129, Springer, Berlin, 2006, pp. 36–44.

[Tzov04]

Tzovaras, D., Nikolakis, G., Fergadis, G., Malasiotis, S., Stavrakis, M., *Design and Implementation of Haptic Virtual Environments for the Training of Visually Impaired*, IEEE Trans. on Neural Systems and Rehabilitation Engineering, Vol. 12, No. 2, June 2004, pp. 266–278.

U

[Unde98]

Underkoffler, J., Ishii, H., *Illuminating light: an optical design tool with a luminous-tangible interface*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'98, ACM Press, New York, 1998, pp. 542-549.

[USIX07]

UsiXML Consortium. UsiXML, a General Purpose XML Compliant User Interface Description Language, UsiXML V1.8, 23 February 2007. Available on-line at: <http://www.usixml.org/index.php?view=page&idpage=6>

V

[Vane06]

Vanacken, D., De Boeck, J., Raymaekers, Ch., Coninx, K., *NiMMiT: A Notation for Modeling Multimodal Interaction Techniques*, Proc. of the Int. Conf. on Computer Graphics Theory and Applications GRAPP'2006 (Setúbal, 25-28 February 2006).

[Vand06]

Vanden Bossche, P., *Développement d'un outil de critique d'interface intelligent : Usability Adviser*, M.Sc. thesis, Université catholique de Louvain, Louvain-la-Neuve, 1 September 2006.

[Vand93]

Vanderdonckt, J., Bodart, F., *Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection*, Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), ACM Press, New York, 1993, pp. 424–429.

[Vand94]

Vanderdonckt, J., *Guide ergonomique des interfaces home-machine*, Presses Universitaires de Namur, Namur, 1994.

[Vand00]

Vanderdonckt, J., *A Small Knowledge-Based System for Selecting Interaction Styles*, Proc. of Int. Workshop on Tools for Working with Guidelines TFWWG'2000 (Biarritz, 7-8 October 2000), Springer-Verlag, London, 2000, pp. 247-262.

[Vand04]

Vanderdonckt, J., Bouillon, L., Chieu, K.C., Trevisan, D., *Model-based Design, Generation, and Evaluation of Virtual User Interfaces*, Proc. of 9th ACM Int. Conf. on 3D Web Tech. Web3D'2004 (Monterey, 5-8 April 2004), ACM Press, New York, 2004, pp. 51–60.

[Vand05]

Vanderdonckt, J., *A MDA-Compliant Environment for Developing User Interfaces of Information Systems*, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31.

References

- [Vand08] Vanderdonckt, J., González-Calleros, J.M., *Task-Driven Plasticity: One Step Forward with UbiDraw*, Proc. of Engineering Interactive Systems HCSE/TAMODIA'2008 (Pisa, 25-26 September 2008), Lecture Notes in Computer Science, Vol. 5247, Springer-Verlag, Berlin, 2008, pp. 181-196.
- [Vand09a] Vanderdonckt, J., Cantera Fonseca, J.M., Diaz Diaz, J.L., González Calleros, J.M., *A Set of Reference Models for Interactive Applications*. NEXOF-RA Investigation Team on Declarative UI Authoring and Context Models. Available on-line at: http://forge.morfeo-project.org/wiki_en/index.php/Interactive_Application_Models
- [Vand09b] Vanderdonckt, J., Guerrero-Garcia, J., González-Calleros, J.M., *A Model-Based Approach for Developing Vectorial User Interfaces*, Proc. of Joint 4th Latin American Conference on Human-Computer Interaction-7th Latin American Web Congress LA-Web/CLIHIC'2009 (Merida, 9-11 November 2009), E. Chavez, E. Furtado, A. Moran (Eds.), IEEE Computer Society Press, Los Alamitos, 2009, pp. 52-59
- [Verj95] Verjans, S. *State-of-the-art for Input Modalities*. Esprit BRA AMODEUS-II Working Paper TM-WP20. 1995. Available on-line at: <http://perswww.kuleuven.be/~u0003438/>.
- [Vitz06] Vitzthum, A., *SSIML/AR: A Visual Language for the Abstract Specification of Augmented Reality User Interfaces*, Proc. of the IEEE Virtual Reality Conference VR'2006 (25-29 March 2006), IEEE Computer Society Press, Los Alamitos, 2006, pp. 135-142.
- [vonG03] von Glasersfeld, E., *An Exposition of Constructivism: Why Some Like it Radical*, The internet encyclopedia of personal construct psychology of the Scientific Reasoning Research Institute, 2003. Available on-line at: <http://www.oikos.org/constructivism.htm>
- W**
- [W3C01] W3C consortium, XML Schema Specification, W3C Recommendation, 2 May 2001. Available on-line at: <http://www.w3.org/XML/schema.html>.
- [W3C04] W3C (2004), Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C recommendation, 16 March 2004, W3C Consortium. Available on-line at: <http://www.w3.org/TR/voicexml20>.
- [W3C03] W3C (2003), Device Independence Principles, W3C Working Group Note, 01 September 2003. W3C Consortium. Available on-line at: <http://www.w3.org/TR/2003/NOTE-di-princ-20030901/>
- [W3C06] W3C (2006), W3C InkML: Digital Ink Markup Language, W3C recommendation, 24 October 2006, W3C consortium. Available on-line at: <http://www.w3.org/2002/mmi/ink>
- [W3C07a] W3C (2007a), Dial: Device Independent Authoring Language, W3C Working Draft, W3C consortium. Available on-line at: <http://www.w3.org/TR/dial/>
- [W3C07b]

References

- W3C (2007b), Content Selection Primer 1.0, W3C Working Draft, W3C consortium. Available on-line at: <http://www.w3.org/TR/cselection-primer/>
- [W3C07c] W3C (2007c), XForms 1.0 (Third Edition), W3C recommendation, 29 October 2007, W3C consortium. Available on-line at: <http://www.w3.org/TR/2007/REC-xforms-20071029/>
- [W3C08] W3C (2008), EMMA: Extensible MultiModal Annotation markup language, W3C Proposed Recommendation, W3C consortium. Available on-line at: <http://www.w3.org/TR/emma>.
- [W3C09] W3C Model-based User Interfaces Incubator Group. Available on-line at http://www.w3.org/2005/Incubator/model-based-ui/wiki/Main_Page
- [W3C95] W3C consortium, *VRML Virtual Reality Modeling Language*, 17 April 1995. Available on-line at: <http://www.w3.org/MarkUp/VRML/>
- [Wall03] Wall, S.A., Brewster, S.A., *Assessing Haptic Properties for Data Representation*, Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'2003 (Fort Lauderdale, 2003), ACM Press, New York, 2003.
- [Wals95a] Walsham, G., *Interpretive Case Studies in IS Research: Nature and Method*, European Journal of Information Systems, Vol. 4, 1995, pp. 74-81.
- [Wals95b] Walsham, G., *The Emergence of Interpretivism in IS Research*, Information Systems Research, Vol. 6, No. 4, 1995, pp. 376-394.
- [Wang05] Wang, S., Poturalski, M., Vronay, D., *Designing a Generalized 3D Carousel View*, Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2005 (Portland, 2-7 April 2005), ACM Press, New York, 2005, pp. 2017-2020.
- [Wate93] Waterworth, J.A., Serra, L., *VR Management Tools: Beyond Spatial Presence*, Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), ACM Press, New York, 1993, pp. 319-320.
- [Web304a] Web3D Consortium, *Information technology — Computer graphics and image processing — Extensible 3D (X3D)*. Available on-line at: <http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/>
- [Web304b] Web3D Consortium. *X3D Frequently Asked Questions*. Available on-line at: <http://www.web3d.org/x3d/faq/>
- [Weli00] Welie, van M., Traetteberg, H., *Interaction Patterns in User Interfaces*, Proc. Seventh Pattern Languages of Programs Conference PLoP'2000 (Allerton Park Monticello, August 2000).
- [Wess03] Wesson, J., Cowley, N.L.O., *Designing with patterns: Possibilities and pitfalls*, Proc. of 2nd Workshop on Software and Usability Cross-Pollination: The Role of Usability Patterns, 2003.

References

- [Wiki05] Wikipedia the free encyclopedia. Available on-line at: http://en.wikipedia.org/wiki/Main_Page.
- [Will01] Willans, J., Harrison, M.D., *A Toolset Supported Approach for Designing and Testing Virtual Environment Interaction Techniques*, International Journal of Human-Computer Studies, Vol. 55, No. 2, August 2001, pp. 45-165.
- [Wils78] Wilson, E.O., *On Human nature*, Harvard University Press, Cambridge, 1978.
- [Wils02] Wilson, J.R., Eastgate, R., D'Cruz, M., *Structured Development of Virtual Environments*, in "Handbook of Virtual Environments: Design, Implementation, and Applications", J. Jacko (Ed.), Lawrence Erlbaum Associates, Mahwah, 2002.
- [Wils97] Wilson, B., *The postmodern paradigm*, in "Instructional development paradigms", C. R. Dills and A. Romiszowski (Eds.), Educational Technology Publications, Englewood Cliffs, 1997. Available on-line at: <http://www.cudenver.edu/~bwilson/postmodern.html>.
- [Wing09] Wingrave, C.A., Laviola, J.J., Bowman, D.A., *A natural, tiered and executable UIDL for 3D user interfaces based on Concept-Oriented Design*, ACM Trans. on Computer-Human Interaction, Vol. 16, No. 4, November 2009, pp. 21-36.
- [Witm98] Witmer, B.G., Singer, M.J., *Measuring Presence in Virtual Environments: A Presence Questionnaire*, Presence, Vol. 7, No. 3, 1998, pp 225-240.
- Y**
- [Ying06] Ying, J., *An approach to Petri net based formal modelling of user interactions from X3D content*, Proc. of the 11th Int. Conf. on 3D Web Technology Web3D'2006 (Columbia, 18-21 April 2006), ACM Press, new York, 2006, pp. 153-157.
- [Yu02] Yu, W., Reid, D., Brewster, S.A., *Web-Based Multimodal Graphs for Visually Impaired People*, Proc. of the 1st Cambridge Workshop on Universal Access and Assistive Technology CWUAAT (Cambridge, 2002), pp.97-108.
- Z**
- [Zaki] Zakiul, S. *Week 15 report on Project 6*. Available on-line at: <http://www.public.asu.edu/~zakiul/vrml/week15/week15.htm>

Appendix A. Transformational Rules

1. Basic nodes capabilities

Node creation. Figure A-1 represents the emptiness of the left hand side providing that no condition is necessary to create the node described in the right hand side.

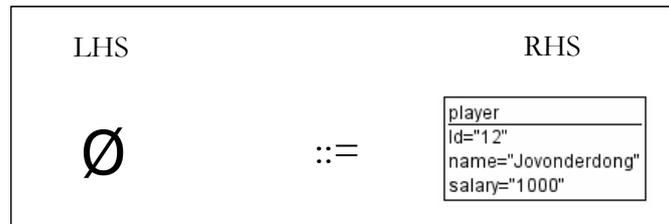


Figure A-1 Creation of a node with attributes

Node modification (identified instance). Figure A-2 shows a rule selecting a specific node on the base of its id attribute and assigns to this node a specific attribute value.

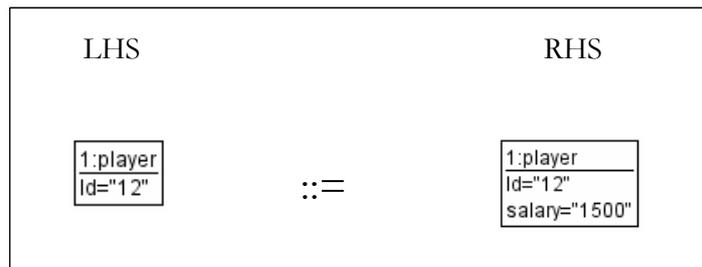


Figure A-2 Node modification (identified instance)

Node modification (unidentified instance). Figure A-3 shows a rule that could be expressed as follows: “for all players that played the match on the 04/06/04, align their salary to 2000”.

Appendix A. Transformational Rules

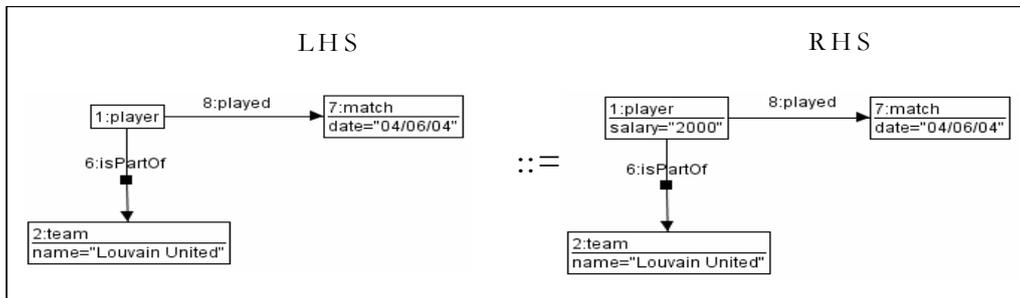


Figure A-3 Node modification unidentified instance

Negative application condition (1). A negative application condition could be added to the preceding rule (Figure A-4). This negative application condition transforms the meaning of the rule into: “for all players that played the match on the 04/06/04, raise their salary to 2000 unless they played the match of the 10/10/03” (this last match was a very bad one!).

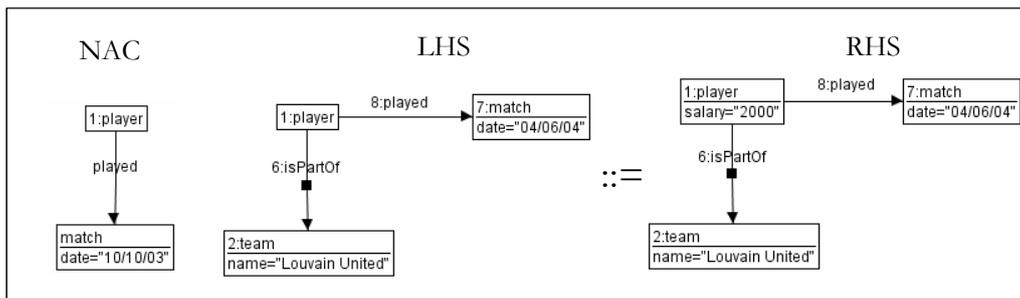


Figure A-4 Negative application condition

Negative application condition for iterative execution of rules (2). Rules that detect patterns on a graph structure and make appropriate modifications depending on the presence of this pattern, is possible for the system that search iteratively for the left hand side. Consequently, there is a risk that the pattern matching algorithm will match several times on the same instances leading to an infinite looping of the execution of the rule. For this purpose a special negative application condition has to be introduced. “NAC2” in Figure A-5, is such an example. It says that the rule should not be applied if the salary of the player equals already “2000”.

Appendix A. Transformational Rules

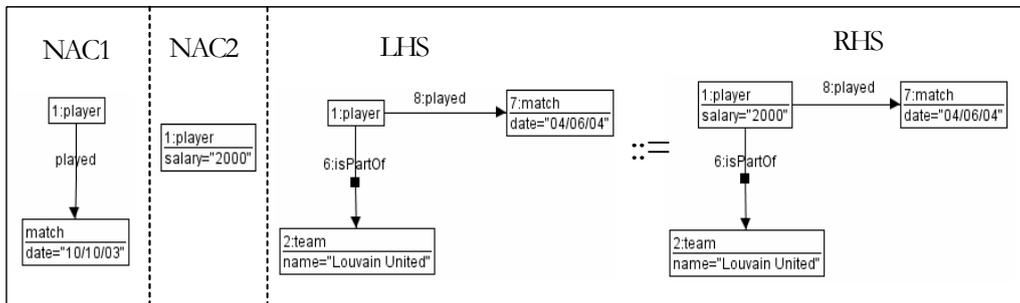


Figure A-5 Negative Application Condition (2)

Rule with variable and variable condition as positive application condition.

Figure A-6 could be expressed as follows: “raise by 500 the salary of all players that played the match of the 04/06/04 only if their salary was inferior to 3000”. This rule illustrates two different mechanisms. A first one consists in the use of a variable in the left hand side, this variable is incremented by a constant in the right hand side (“ $x = x + 500$ ”). A second one consists in the use of a positive application condition that compares the value of a variable with a constant (note that x could have been compared with another variable).

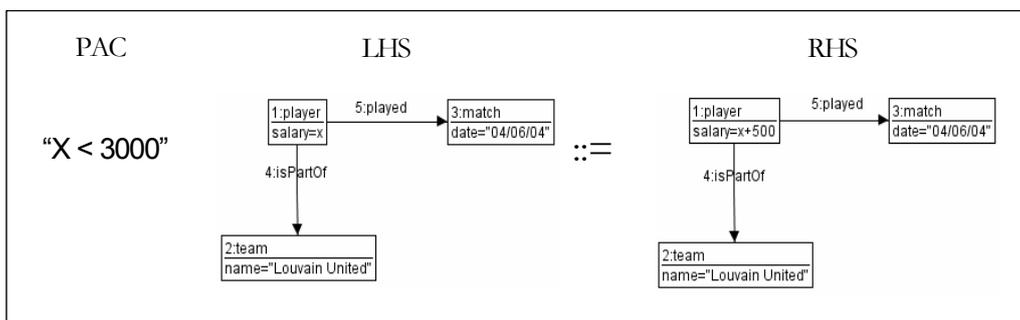


Figure A-6 Rule with variable and positive application condition

Transfer of an attribute value. Figure A-7 illustrates a very altruistic rule, which may be expressed as follows: “If two players of a same team are friends and one earns more than the other, then align their salaries”. Here the value of a variable is transferred from one node (the richest player) to another one (the poor friend).

Appendix A. Transformational Rules

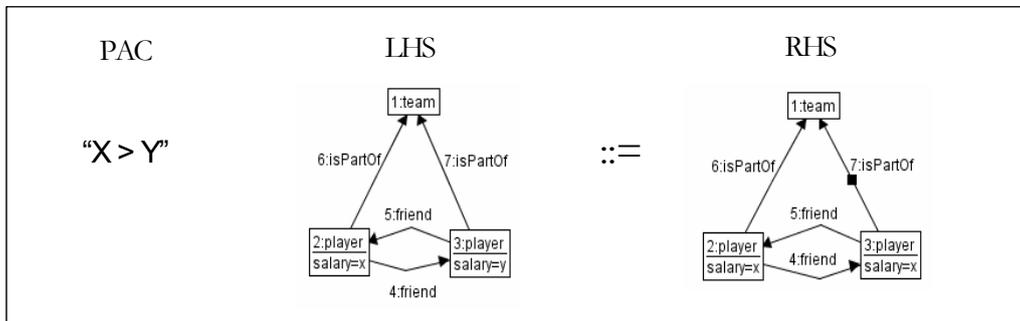


Figure A-7 Transfer of an attribute value

Edge creation. Figure A-8 illustrates a rule that could be expressed as follows “All players of Louvain United with a salary greater than 3000 should be assigned to the match of the 04/06/04” (It will be a tough match !).

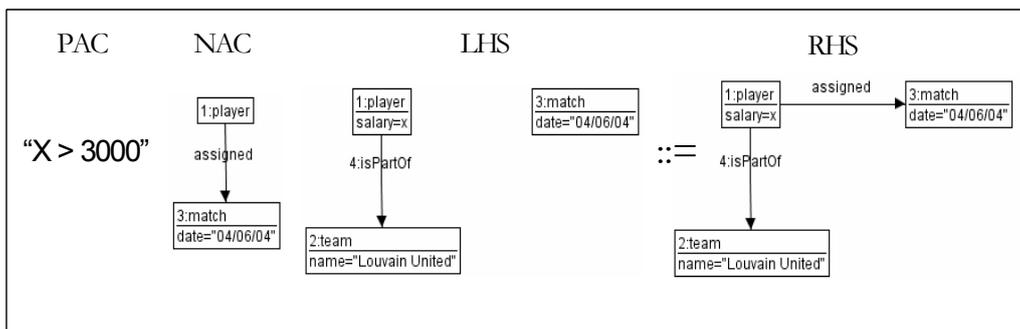


Figure A-8 Edge creation

Node deletion. Figure A-9 shows the most delicate operations of all: node deletion. Indeed, the problem with node deletion is that they raise the question of dangling edges. We adopt a very clear policy regarding this problem: all edges pointing to or originating from a deleted node should be erased. In other words, no dangling edges are allowed.

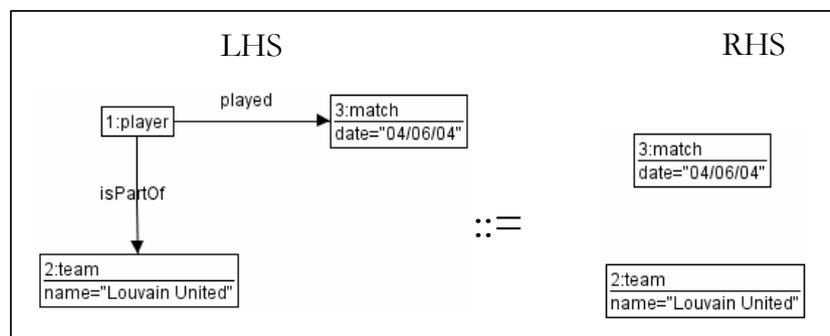
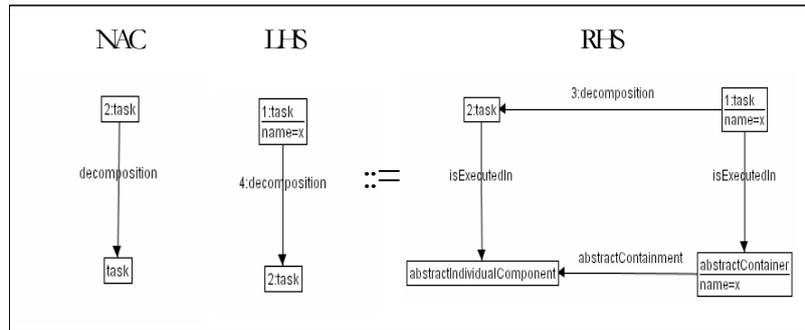


Figure A-9 Node deletion

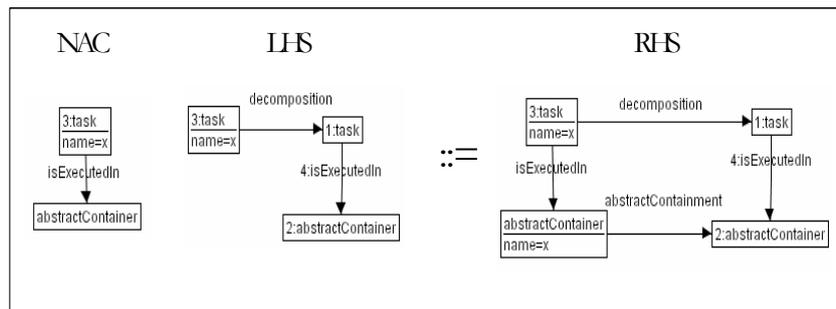
Appendix A. Transformational Rules

Rule 1: For each leaf task of a task tree, **create an Abstract Individual Component**. For each task, parent of a leaf task, create an Abstract Container. Link the abstract container and the Abstract Individual Element by a containment relationship.



Rule 1 Creation of abstract individual components derived from task model leaves

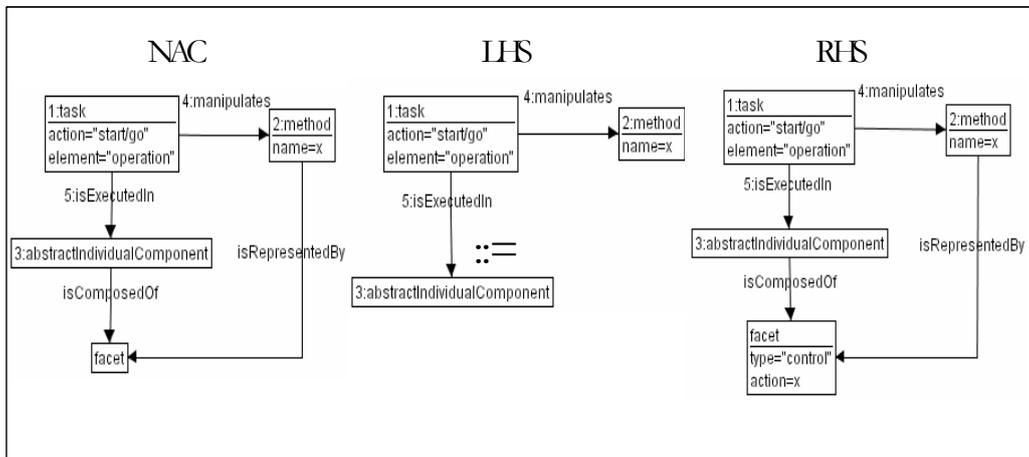
Rule 2: **Create an Abstract Container** structure parallel to the task decomposition structure.



Rule 2 Creation of abstract containers derived from task model structure

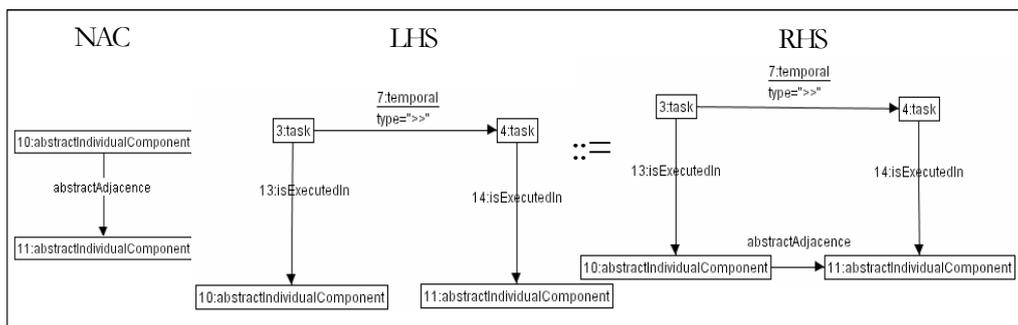
Rule 3: for each **abstract individual element mapped onto a task** such that the tasks nature consists of the activation of a method and this task is mapped onto a class, assign to the abstract individual component an action facet that activates the mapped method.

Appendix A. Transformational Rules



Rule 3 Creation of a facet for an abstract individual component derived from task action type

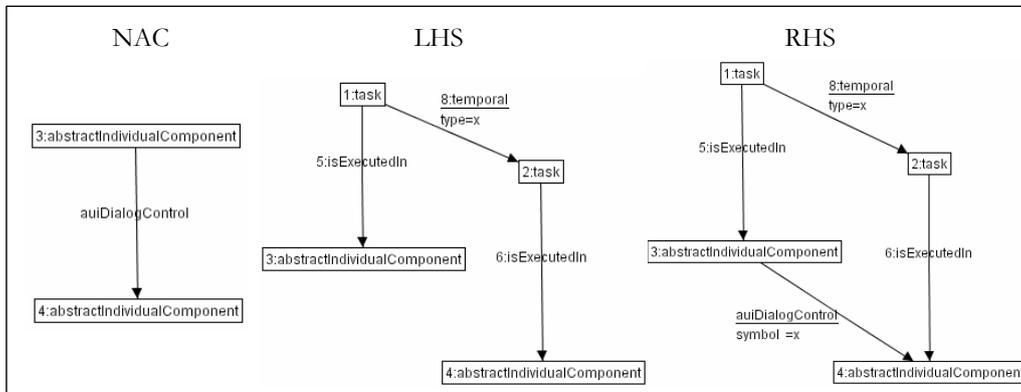
Rule 4: for every couple of AIC mapped onto sister tasks that are sequential “>>”, create a relationship of type “abstractAdjacence” between these AIOs.



Rule 4 A sequentialisation of abstract individual component derived from task temporal relationships

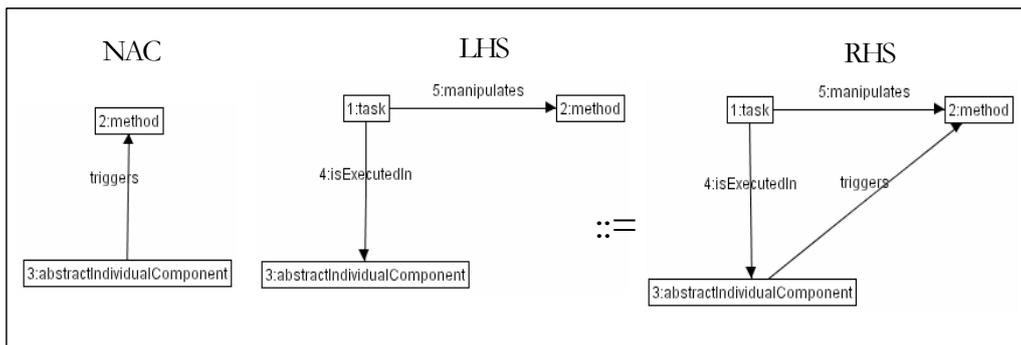
Rule 5: for each couple of sister tasks mapped onto AICs, define a dialog control relationship between these AIC that has the same semantic as the temporal relationship.

Appendix A. Transformational Rules



Rule 5 Abstract Dialog Derivation from Task Model

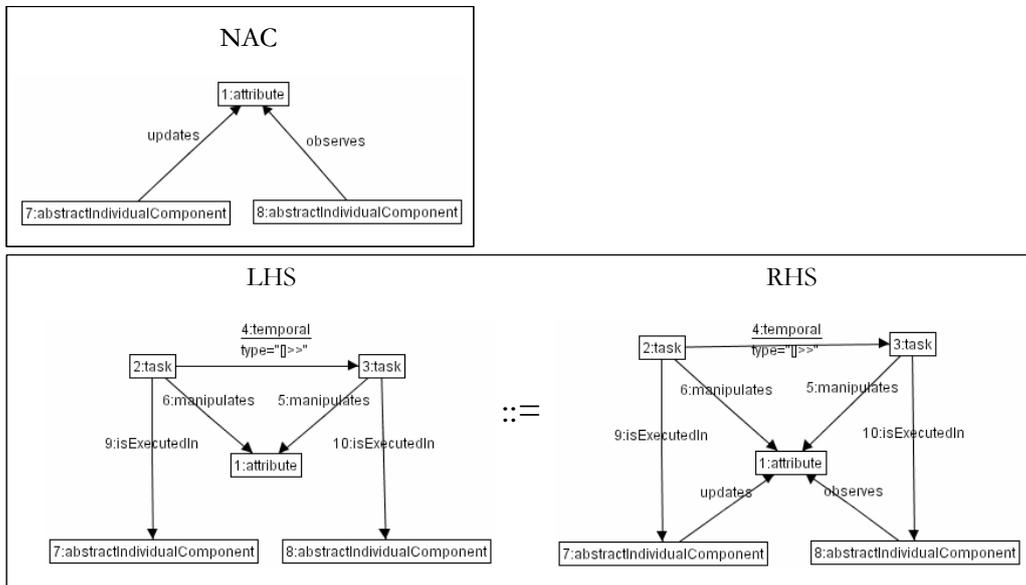
Rule 6: for each **task that manipulates a method**, the AIC that represents this task triggers the method.



Rule 6 Deriving triggering relationships from task domain mappings

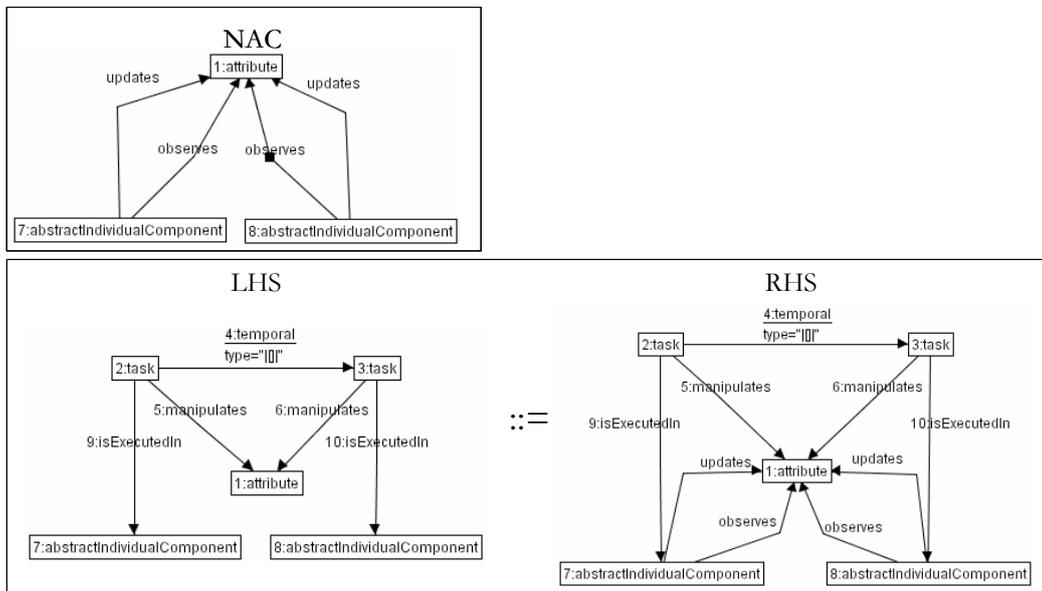
Rule 7: if two sister tasks manipulate a same attribute and are temporally constrained with a “sequence with information passing” relationship, each of these tasks being mapped onto an AIC, then the AIC that is mapped with the first task updates the attribute manipulated by the tasks. The second AIC observes this attribute.

Appendix A. Transformational Rules



Rule 7 Derivation of Updates and observes structure on the base of a task relationship of sequential information passing

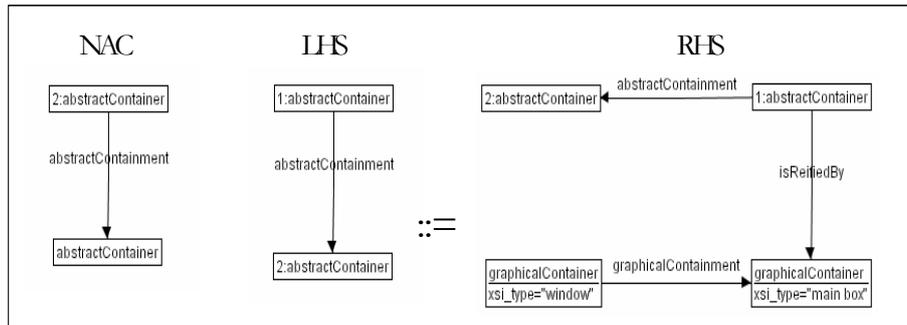
Rule 8: if two sister tasks manipulate a same attribute and are temporally constrained with a “concurrent information passing” relationship, and each of these tasks is mapped onto an AIC, then both AIC observe and update the attribute that is manipulated by the tasks.



Rule 8 Derivation of Updates and Observes structure on the basis of a task relationship of concurrent information passing

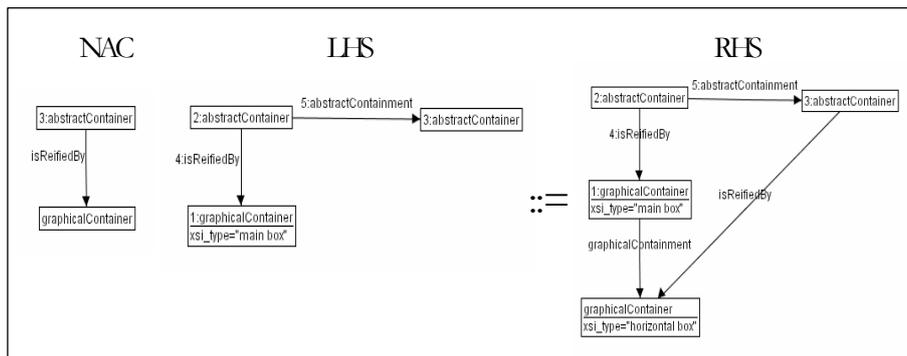
Appendix A. Transformational Rules

Rule 9: Each **abstract container at level “leaf-1”** is transformed into a **window**. Note that an abstract container is always reified into a, so called, box at the concrete level. This box is then embedded into a window.



Rule 9 A creation of windows derived from containment relationships at the abstract level

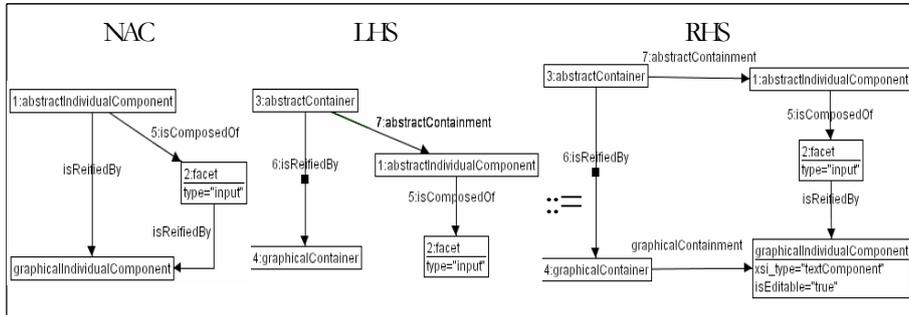
Rule 10: each abstract container contained into an abstract container that was reified into a window is transformed into an horizontal box and embedded into the window.



Rule 10 A generation of window structure derived from containment relationship at the abstract level

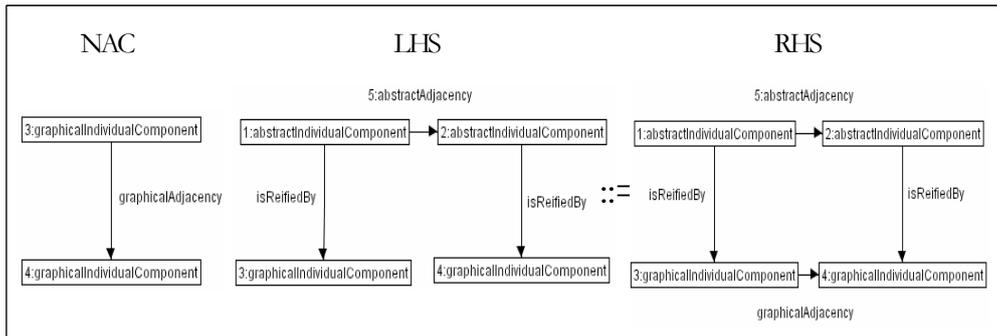
Rule 11: each input facet of an abstract individual component is reified by a graphical individual component (a type of concrete individual component) of type “editable text component” (i.e., a text box).

Appendix A. Transformational Rules



Rule 11 Creation of an editable text component (i.e., an input field) derived from facets type of abstract components

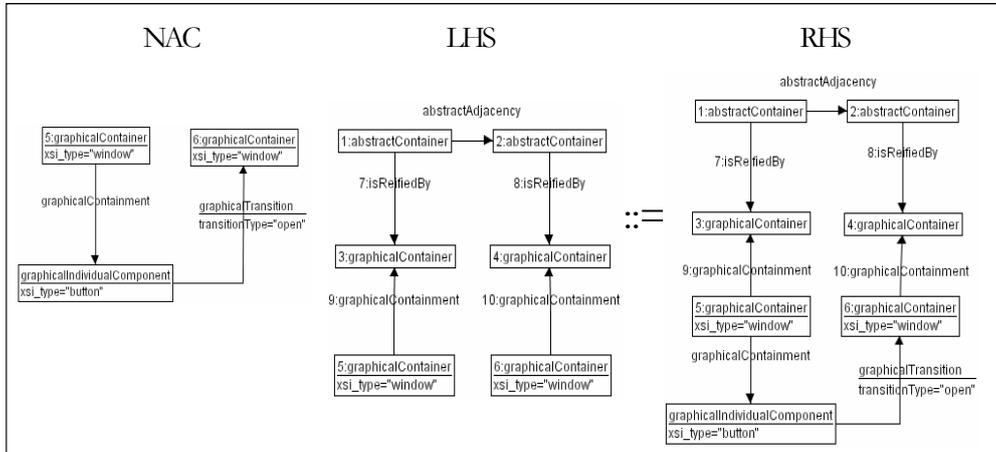
Rule 12: for each couple of abstract individual components related by an “abstractAdjacency” relationship and reified into concrete individual components, generate a “concreteAdjacency” relationship between the concrete individual components.



Rule 12 A placement of graphical individual components derived from spatio-temporal relationships at the abstract level

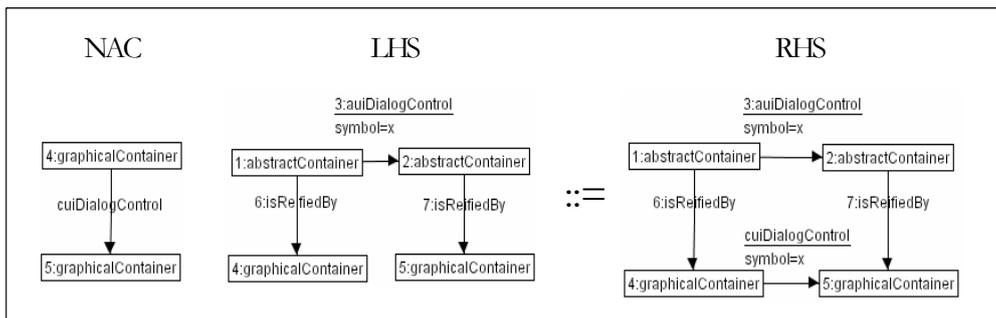
Rule 13: for each container related to another container belonging to different windows, and their respective abstract container being related by a “is before relationship”, generate a navigation button in source container pointing to the window of target container.

Appendix A. Transformational Rules



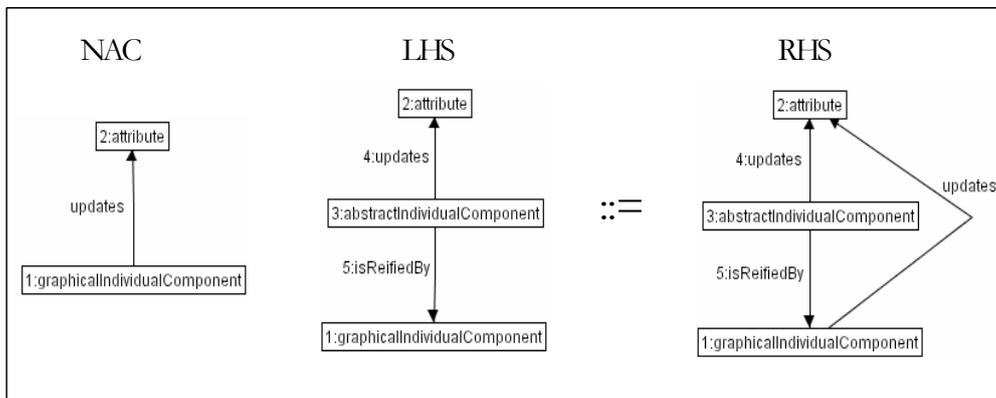
Rule 13 A window navigation definition derived from container adjacency relationships

Rule 14: for each couple of abstract container with a dialog control relationship, transpose this relationship to the couple of concrete containers that reify them.



Rule 14 Derivation of the concrete dialog from abstract dialog

Rule 15: for each AIC updating a domain concept, if a CIC reifies this AIC then the CIC updates this same domain concept.



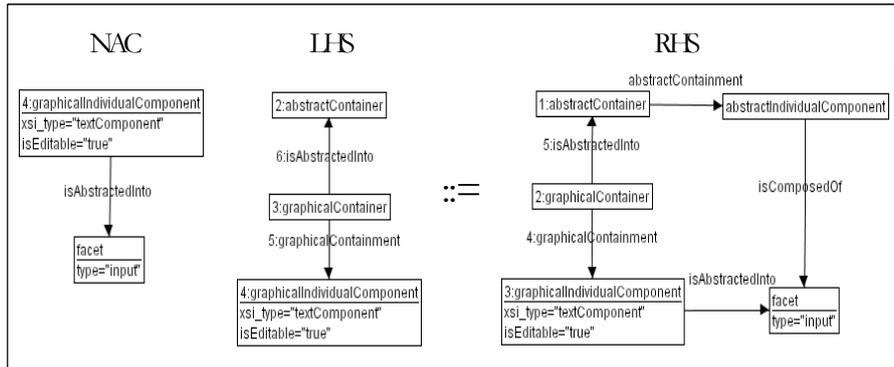
Rule 15 Transposition of update relationship

2. From Concrete User Interface to Code Rules

Step T3 consists in code generation from a CUI. Code generation techniques for UIs is a very well known topic. [Czar00] presents a state-of-the art of model to code techniques (e.g., visitor-based approach and template based approach). Scientific results for this transformation have been shown in systems issued from research like: Janus [Balze95], Trident [Boda95], Modi-D [Puer97] or from commercial world e.g., Genova [Geno04] or Oliva Nova [Moli02]. The present work does not particularly contribute to this area although several tools have been developed to provide code generation support from the concrete user interface level.

3. Reverse Engineering Rules

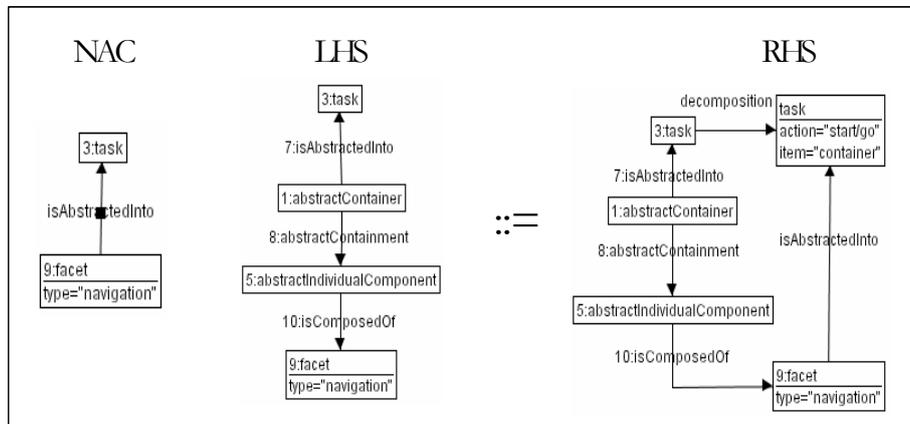
Rule 16: for each editable graphical individual component, create an abstract individual component equipped with an input facet.



Rule 16 Creation of a facet at the abstract level derived from a type analysis of graphical individual components

Rule 17: for each abstract individual component equipped with a navigation facet create a task with action type “start/go” on an item of type “element”.

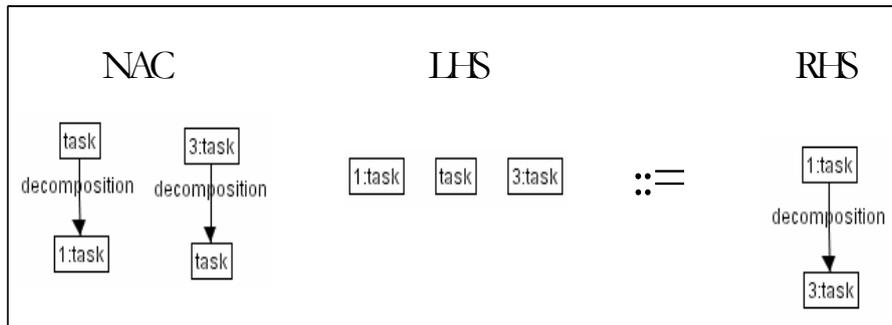
Appendix A. Transformational Rules



Rule 17 Definition of task action types derived from an analysis of facets at the abstract level

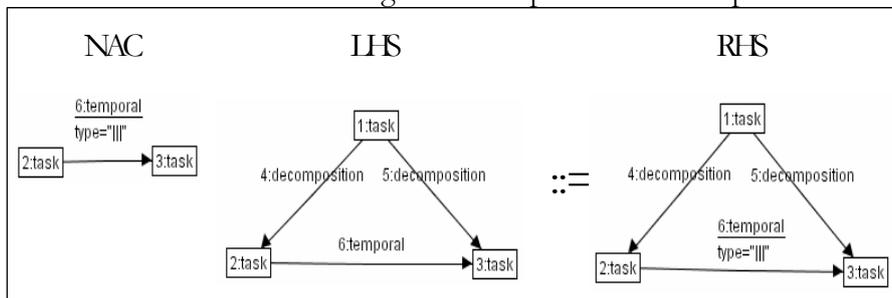
4. Adaptation to context change

Rule 18: (1) erases each intermediary task (i.e., non-leaf and non-root tasks). (2) attaches every leaf task to the root.



Rule 18 Flattening of a task tree structure

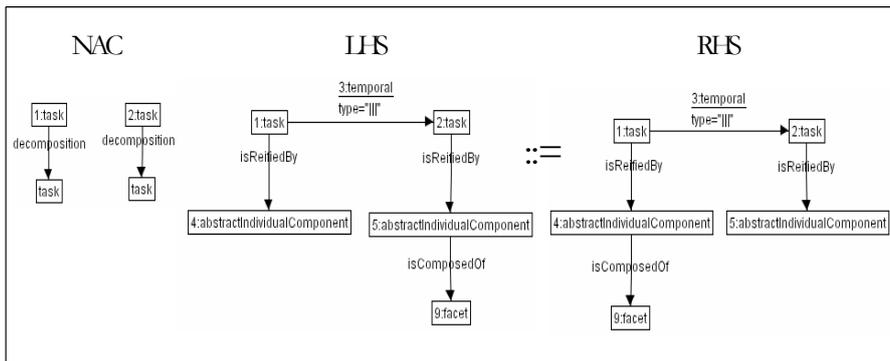
Rule 19: for each sister tasks change their temporal relationship into concurrent.



Rule 19 Transforming all temporal relationship to concurrent

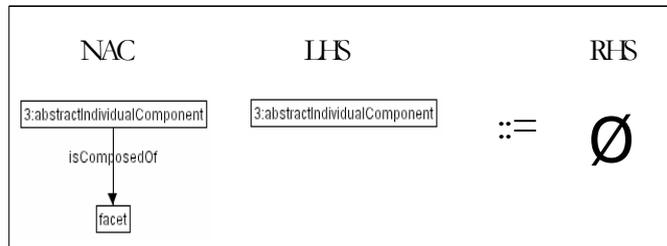
5. Step: From Abstract User Interface to Abstract User Interface

Rule 20: for each pair of abstract individual component mapped onto concurrent tasks, transfer all facets of the abstract individual component that is mapped onto the task target of the concurrency relationship, to the other abstract individual component.



Rule 20 A merging of facets of abstract individual components

Rule 21: erase all abstract individual components that have no facets left.



Rule 21 Erasing abstract individual components with no facets left

6. Step: From Concrete User Interface to Concrete User Interface

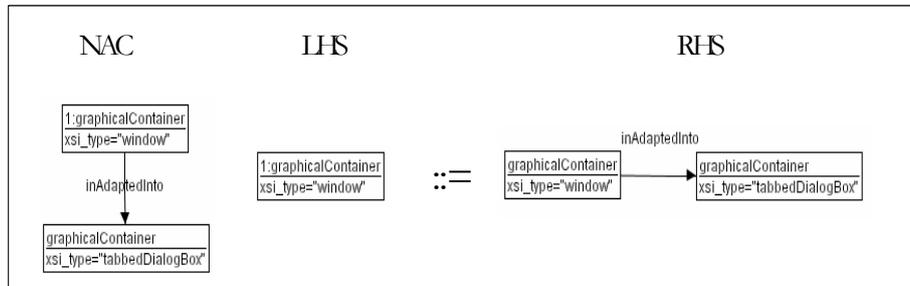
Adaptation at the concrete level is illustrated by several development sub-steps: container type modification (called concrete container re-formation), modification of the types of concrete individual components (called concrete individual components re-selection), layout modification (layout re-shuffling), or navigation re-definition. Examples for these first three adaptation types are given hereafter.

7. Sub-step: Concrete container re-formation

Concrete container Re-Formation may cover situations like container type transformation (e.g., a window is transformed into a tabbed dialog box), container system modification (e.g., a system of windows is merged into a single window).

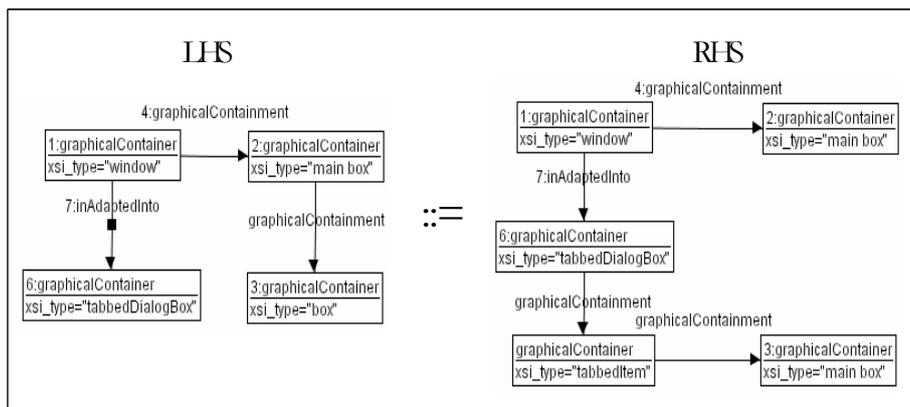
Appendix A. Transformational Rules

Rule 22: each window is selected and mapped onto a newly created tabbed dialog box.



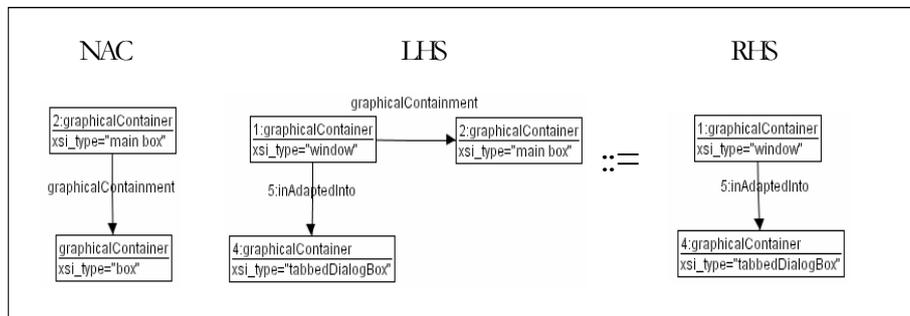
Rule 22 Initializing of the adaptation process by creating graphical component to adapt into

Rule 23: transfers every first level box of the window to adapt it into a tabbed item composing a tabbed dialog box.



Rule 23 Creation of a tabbed item and transfer of the content of the adapted window

Rule 254: cleans up the specification of remaining empty main boxes.

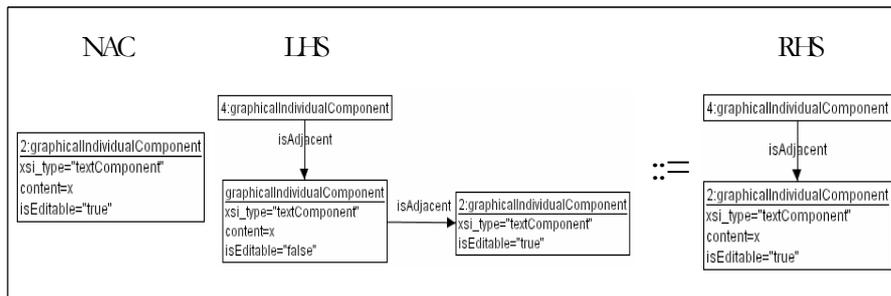


Rule 24 Deletion of unnecessary containers

8. Sub-step: Concrete individual component re-selection

Re-selection transformations adapt individual component into other individual components.

Rule 25: for each couple of adjacent editable text component and non-editable text component. Erase the editable text component and transfer its content into the non-editable text component (unless a content has already been transferred).

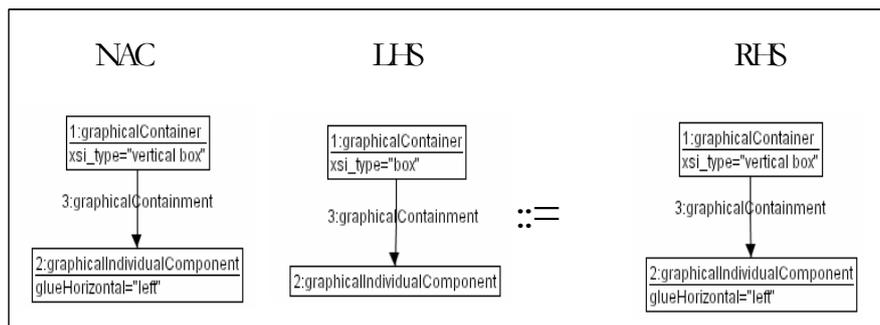


Rule 25 Merging of a non-editable text component (e.g., a label) and an editable text component (e.g., an input field) into one single editable text component

9. Sub-step: Layout re-shuffling

A layout at the concrete level is specified with horizontal and vertical boxes. An elements contained into a box may be glued to an edge of this box.

Rule 26: each box is transformed into a vertical box and every individual component is glued to left.



Rule 26 Squeezing of a layout structure to display vertically

Appendix B. UsiXML Ontology for User Interface Specification

The first description of Limbourg [Limb04c] refers to the separation of concerns, inspired by Dijkstra [Dijk76]: “This principle states that the different aspects of a problem should be isolated from one to each other. Separation of concerns allows studying fractions of a matter in an independent manner while modularizing this matter. A concern gathers properties relevant to one perspective that can be maintained on an artefact”

In [Limb04b] they introduce UsiXML language to handle the concepts defined in their ontology of UIs. This language is structured according to four basic levels of abstractions defined by the Cameleon reference framework [Calv03] (see Figure B-1).

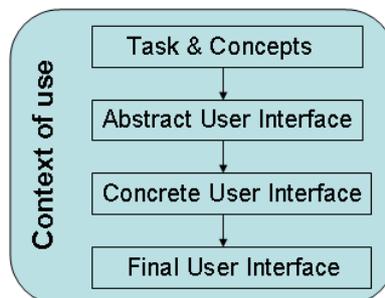


Figure B-1 The Cameleon reference framework for multi-target UIs

The description of four levels of their approach is described as follows:

- At the top level is the Task & Concepts level that describes the various interactive tasks to be carried out by the end user and the domain objects that are manipulated by these tasks. These objects are considered as instances of classes representing the concepts.
- An Abstract UI (AUI) provides a UI definition that is independent of any modality of interaction (e.g., graphical interaction, vocal interaction, etc.).

Appendix B. UsiXML Ontology for User Interface Specification

- As an AUI does not refer to any particular modality, we do not know yet how this abstract description will be concretized: graphical, vocal or multimodal. This is achieved in the next level.
- The Concrete UI (CUI) concretizes an AUI for a given context of use into Concrete Interaction Objects (CIOs) so as to define layout and/or interface navigation of 2D graphical widgets and/or vocal widgets. Any CUI is composed of CIOs, which realize an abstraction of widgets sets found in popular graphical and vocal toolkits. A CIO is defined as an entity that users can perceive and/or manipulate (e.g., push button, text field, check box, vocal output, vocal input, vocal menu). The CUI abstracts a Final UI in a definition that is independent of programming toolkit peculiarities.
- The Final UI (FUI) is the operational UI, i.e. any UI running on a particular computing platform either by interpretation (e.g. through a Web browser) or by execution (e.g., after the compilation of code in an interactive development environment).

As depicted in Figure B-1, the Context of use surrounds the different levels. This context of use describes all the entities that may influence how the user's task is carrying out with the future UI. It takes into account three relevant aspects, each aspect having its own associated attributes contained in a separate model: user type (e.g., experience with device and/or system, task motivation), computing platform type (e.g., desktop, PocketPC, PDA, GSM), and physical environment type (e.g., lighting level, stress level, noise level). These attributes initiate transformations that are applicable depending on the current context of use.

Finally, in order to map different elements belonging to the models described above, UsiXML provides the designer with a set of pre-defined relationships called mappings.

1. Task Model

A *task model* describes the various tasks to be carried out by a user in interaction with an interactive system. The task model used in this methodology is similar as the proposed in [USIX06], which is an extended version of ConcurTaskTree (CTT) [Pate97], selected as it represents user's tasks along with their logical and temporal ordering.

The proposed task model is composed of *tasks* and *task relationships*, see Figure B-2. A task *frequency* attribute is an assessment of the relative frequency of execution of a task. Task frequency is evaluated on a scale from 1 to 5. A task *importance* attribute assesses the relative importance of a task with respect to main user's goals.

Appendix B. UsiXML Ontology for User Interface Specification

Task importance is evaluated on a scale from 1 to 5. A value of 1 means that a task has a low importance, 5 means that a task is very important. Tasks have been described with a *name*, and a *type*. Task *type*, similarly as proposed by [Pate97], may be:

User tasks are notably useful to predict a task execution time, as is the user responsible on executing them. A user task refers to a cognitive action like taking a decision, or acquiring information. An interactive task involves an active interaction of the user with the system (e.g., selecting a value, browsing a collection of items). A system task is an action that is performed by the system (e.g., check a credit card number, display a banner). An abstract task is an intermediary construct allowing a grouping of tasks of different types.

Task relationships are:

- Enabling (T1 has to be finished in order to initiate T2)
- Non-deterministic choice (Once one task is finished the other cannot be accomplished anymore)
- Deterministic Choice (Once one task is initiated, the other cannot be accomplished anymore)
- Parallelism (T1 is interleaved with T2).
- Sequential independence (Is equivalent to (T1 >> T2) OR (T2 >> T1))
- Deactivation (T2 may interrupt T1 before the termination of T1; T1 cannot be resumed after T2 has terminated.)
- Suspend/Resume (T2 may interrupt T1 before the termination of T1. Once T2 is finished, T1 may be resumed.)
- Enabling with information passing. Task T1 has to be finished in order to initiate task T2 and T2 is synchronized with T1 on some piece of data.
- Parallelism with information passing. Task T1 is interleaved with task T2 while they synchronize on some data.
- Task Iteration (*|n). Task T can be iterated an infinite number of times *, or n times
- Optional tasks. Task T is optional.

Several additional constraints may be formulated on the consistency of a task model:

Appendix B. UsiXML Ontology for User Interface Specification

- There exists a maximum of one binary (i.e., temporal or decomposition) relationship between two tasks.
- If a task is decomposed into another task then this last task must have a brother task.
- There is only one root task. This means that there is only one element with no decomposition relationship pointing to it.

2. Domain Model

A *domain model* describes the real-world concepts, and their interactions as understood by users and the operations that are possible on these concepts [DSou99]. We selected UML class diagrams as the basis of expression for our domain model. We considered UML class diagrams as Extended Entity Relationship model (EER) [Teor86]. The main reason for this choice is that UML has become a lingua franca in the domain of software engineering and is widely used in industrial practice [Limb04c].

We rely on [USIX06] domain meta-model, shown in Figure B-3. This UML class diagram shows the features added to the initial UML standard in order to better tackle the problem of transformational development of UIs. For instance, the domain n of values attached to attributes is described with a richer precision in order to allow widget selection (e.g., enumerated domains can be described extensively).

From [USIX06] Domain model concepts are:

- **domainClass**. Classes describe the characteristics (attributes and methods.) of a set of objects sharing a set of common properties.
- **Attribute**. Attributes enable a description of a particular feature of a class.
- The type of an attribute refers to common data types found in most programming language i.e., Boolean, char, string, integer, float. The type attribute may also make reference to an object type, such as the vector required to denote a 3D colour.
- The cardinality of an attribute indicates the number of values an attribute may be associated with. The cardinality can be specified by providing two integers: a minimal cardinality and a maximal cardinality.

An original typology allows characterizing a type of *domain* for an attribute. Indeed, **attributeDomainCharacterization** takes the value of: interval, continuous interval, discrete interval, linear interval, circular interval, set[n] (where n is the number of possible values in an attribute domain). When used in combination

Appendix B. UsiXML Ontology for User Interface Specification

with a task model, this typology helps to map domain attributes to a type of interaction object by which it will be rendered [Limb04c]. For instance, a “choose element” task on an attribute with a circular interval enables the derivation of a (multi-state) toggle button.

Methods (in this context) are presences which are called either by objects of the domain or by user interface components. Methods manipulate object’s attributes. Methods are, here, described with their signature i.e., with their name, type, and parameters.

Objects are instances of a class. An object is composed of attribute instances which may have values and define the state of an object.

domain class relationships describe various types of relationships between classes. They can be classified in three types: *generalization*, *aggregation*, *usage*, *materialization*, *instanciation* and *ad hoc*. Class relationships are described with several attributes enabling the specification of role names and cardinalities.

3. Abstract User Interface Model

From [Limb04c] *Abstract User Interface (AUI) model* is defined as “a user interface model that represents a canonical expression of the renderings and manipulation of the domain concepts and functions in a way that is as independent as possible from modalities and computing platform specificities”.

We rely to a similar AUI (Figure B-4). The AUI is populated by *Abstract Interaction Objects (AIO)* and *abstract user interface relationships*. These concepts constitute a vocabulary that is independent of the modality and the computing resources for which a system is targeted at.

A *modality* (also called interaction technique) can be defined more precisely, after [Niga95], as the coupling of a physical device d with an interaction language L : $\langle d, L \rangle$. Our language supports, at the concrete level, two modalities: speech (i.e. *auditory*) input and output and graphic (i.e., *graphical*) input and output. This support will be extended at the graphical level that considered 2D UI.

Abstract Interaction Object (AIO) may be of two types *Abstract Individual Components (AIC)* and *Abstract Containers (AC)*.

An *Abstract Individual Component (AIC)* is an abstraction that allows the description of interaction objects in a way that is independent of the modality in which it will be rendered in the physical world. An AIC may be composed of multiple facets. Each facet describes a particular function an AIC may endorse in the physical world. Four main facets are identified:

Appendix B. UsiXML Ontology for User Interface Specification

- An *input* facet describes the input action supported by an AIC.
- An *output* facet describes what data may be presented to the user by an AIC.
- A *navigation* facet describes the possible container transition a particular AIC may enable.
- A *control* facet describes the links between an AIC and system functions i.e., methods from the domain model when existing.

A single AIC may assume several facets at the same time. The AIO that reifies this multi-facetted AIO will assume all those ‘functionalities’. For instance, a CIO may display an output while accepting an input from a user, ensure a transition between windows and trigger a method defined in the domain model.

An *Abstract Container* (AC) is an entity allowing a logical grouping of other abstract containers or abstract individual components. AC are said to support the execution of a set of logically/semantically connected tasks. Actually AC may be reified, at the concrete level, into one or more graphical containers like windows, dialog boxes, layout boxes or time slots in the case of auditory user interfaces. However there is no concretization of these objects for 3DUIs.

Abstract User Interface Relationships (AUI relationship) are relationships that can be drawn between abstract interaction objects of all kinds. Five types of abstract relationships may be defined at this level:

- *Decomposition* relationship allows specifying a hierarchical structure of abstract containers and abstract individual components.
- *AbstractAdjacency* relationship indicates that two AIO are logically adjacent.
- *Spatio-temporal* relationship allows a specification of a very precise layout in time or space in a way that is independent of any modality.
- *Dialog control* relationship allows a specification of a flow of control between the abstract interaction objects.
- *Mutual emphasis* relationship allows specifying that two components should be somehow differentiated at the concrete level. This relationship may be useful in a user interface where the probability of confusing two UI elements is high (e.g., in an airplane cockpit, a field displaying the angular speed and the absolute speed).

4. Concrete User Interface Model

The Concrete User Interface Model (CUI) represents a concretization of an AUI model. A CUI is populated by Concrete Interaction Objects and Concrete User Interface relationships between them. The CUI model is a UI model allowing a

Appendix B. UsiXML Ontology for User Interface Specification

specification of an appearance and behaviour of a UI with elements that can be perceived by users.

By definition, a CUI is modality dependent as any CUI instance refers to the interaction modalities that have been selected for this UI. In contrast to its modality dependence, a CUI remains toolkit independent as no CUI instance does refer to any physical element (i.e., toolkit elements or widget) of the computing platform. Nonetheless, a CUI description can be detailed enough to allow a complete rendering of a user interface [Limb04c].

A CUI model is composed of *Concrete Interaction Objects* (CIO) and *cui relationships*. A Concrete Interaction Object (CIO) is defined as an entity that users can perceive and/or manipulate (e.g., a push button, a list box, a check box, a sound). The actual specification realizes an abstraction of widget sets found in popular toolkits: 2D graphical (Java Swing, HTML 4.01, Flash) and auditory (earcons and VoiceXML 2.0). In other words, CIOs allows an expression of UI elements that is independent of their actual rendering. Our target is to extend this representation to 3DUIs, including languages such VRML, X3D or JAVA 3D.

Graphical and auditory CIOs are further decomposed into containers and individual components. We have just summarized the main characteristics of the actual model more information can be found in the [USIX06] documentation. More emphasis will be dedicated on the extension.

Graphical containers (GC) correspond to classical and common 2D UIs containers. Attributes used are as abstract as possible in order to respect the independence on implementation.

Graphical Individual Components (*GIC*). Text components are differentiated in two types, for input (an input field, a password field, a multi-line input field) and output (a label, a complex textual output as a rtf file) purposes.

Vocal Concrete interaction objects. *Vocal Containers* represent a logical grouping of other auditory containers or auditory individual components. *Vocal individual components* are of five types: auditory output which may consist in music, voice or a simple “earcon” (i.e., an auditory icon), auditory input which is a mere time slot allowing the user to provide an auditory input using her voice, or any other physical device able to produce sound, vocal navigation (Specifies a transition to another vocalForm), break (Interrupts the execution of the current vocalContainer) and exit (Terminates the execution of the vocal interface).

Appendix B. UsiXML Ontology for User Interface Specification

Similarly to Concrete Interaction Objects they are divided into *vocal relationships* and *graphical relationships*. *Dialog control relationship* can be defined between both types of interaction objects [USIX06].

Vocal relationships are of three types: *vocal transition* that enables to specify a transition between two auditory containers; *vocalAdjacency* that indicates a time adjacency between two auditory components; and *vocalContainment* that allows adding or deleting *vocalIndividualComponets* from a *vocalContainer*.

Graphical relationships are of five types: *Graphical transition* specifies navigation links between the different containers populating the UI, *alignment* that may also be specified among any individual component belonging to the same window, *adjacency* indicates that two components are topologically adjacent, *emphasis* enables to specify that two or more graphicalIndividualComponents must be differentiated in some way (e.g., with different colour attributes) and *containment* analog to the vocal containment, allows to specify that a graphicalContainer embeds one or more graphicalContainers or one or more graphicalIndividualComponents.

Dialog control allows a specification of a flow of control between the concrete interaction objects. As so a dialog control may be specified independently of a task model. LOTOS operators are used for this purpose. For instance a relationship `CIC1.EnterCountry []> CIC2.EnterProvince`, indicates that `CIC2` cannot be initiated while `CIC1` is not terminated and that `CIC1` has provided a value for the data on which the two component synchronize with.

Any CIO may be associated with any number of *behaviours*. A *behaviour* is the description of an event-response mechanism that results in a system state change. The specification of behaviour may be decomposed into three types of elements: an *event*, a *condition*, and an *action* [USIX06].

An *event* is a description of a run-time occurrence that triggers an action. They consist of any system event (i.e., issued from a process belonging to the domain), user interface event (i.e., issued in the context of the user interface). A limitation on the events is that they cannot make any reference to coordinates, which is imperative in 3D event handling. Events can be composed into more complex event expressions using a subset of the LOTOS operators introduced earlier. However, as it is not part of the language, the behaviour description is straightforward from the actual [USIX06] specification.

A *condition* is the expression of a state that has to hold true before (pre-condition) or after (post-condition) an action is performed. A condition may be positive or negative. An *action* is a process that results in a state change in the system. An action can be of three types: a *method call*, a *transformation system*, or a *transition*.

Appendix B. UsiXML Ontology for User Interface Specification

A method call is a call to a method that is external to the UI. If a domain model exists, all method calls must reference a method belonging to this model. A method call is normally specified with the name of the method (under the form `Class.methodName`), but other referencing techniques are not forbidden. The method call parameters can be specified by making a reference to the value of a property of an object belonging to the CUI.

A *transformation system* is the expression of any property change at the UI level. We use a mechanism to specify property changes on the UI. This mechanism is similar to the one that will be introduced in Chapter 4. To avoid too much forward reference, it can be said that a transformation system can be explained as follows: when a pattern is found in CUI specification, changes should occur on the elements matching the pattern. A transformation system might be, for instance, “when a green button is found in the specification, change the colour property of this button to red” or “For all text components belonging to the main window, increase their font by a factor of 2”.

A *transition*, also called *navigation*, is a description of a change in the container’s visibility property of a user interface system. A transition has a source (a navigation individual component) and a target (generally a container). Depending on the type of modality, transitions may be of different types (see above in this Section).

5. Context Model

A context model is a model describing the three aspects of a context of use in which an end user is carrying out an interactive task with a specific computing platform in a given surrounding environment [Thev01]. In [USIX06] the context model is composed of *context* and the *plasticity model set*. The *context* is defined as a triple of the form $\langle e, p, u \rangle$ where e is an element of the environments set considered for the interactive system, p is an element of the platforms set considered for the interactive system and u is an element of the users set for the interactive system. The *plasticity model set* composed of plasticity domains, which defines a sub-area in a specified context (itself included in the physical space: user, environment, platform) where a specified AIO/CIO will be represented such as a specified form. Details of the model can be found in [USIX06] documentation.

The environmental model, which is part of the context model, relies on the assumption that the user interact in the physical world. The actual describes any property of interest of the “physical” environment where the user is using the UI on the computing platform to accomplish her interactive tasks. Such attributes may be physical (e.g., lighting conditions), psychological (e.g., level of stress), and organizational (e.g., location and role definition in the organization chart).

6. Inter-Model Relationships

The concepts described in the ontology require a way to be interconnected. Model integration is a well-known issue in transformation driven development of UI [Puer99]. This problem was sorted with the creation of a set of pre-defined relationships allowing a mapping of elements from heterogeneous models and viewpoints. Several advantages were identified in [Limb04c] such as: the derivation of the system architecture (mappings between domain and CUI/AUI models), for traceability in the development cycle (reification, abstraction and translation), for addressing context sensitive issues (has context), for dialog control issues, for improving the preciseness of model derivation heuristics.

The *intermodel relationship* is any type of relationship established between one or many source models and one or many target models [USIX06]. A typical *inter-ModelRelationship* is established between one source model and one target model, but it can be easily imagined that such a relationship can start from one source model to many target models, but from many source models to many target models.

An *interModelRelationship* is the super class of all possible relationships between models and elements of models. Consists of: one to many sources, one to many targets and the source should not necessarily be different from the target.

Several relationships [USIX06] can be defined to explicit the relationships between the domain model and the UI models (both abstract and concrete):

Observes is a mapping between any UI component (at abstract or concrete level) and a domain attribute or instantiated attribute (at run time). Observes enables to specify that a UI component observes a value from the related domain concept. This mapping may be interpreted as follows: the content of a UI object must be synchronized when:

- A mapped attribute is modified. The new state resulting from this modification should be presented on the UI (the notion of view could be of interest here).
- A mapped method is executed. Its output parameters are displayed on the UI.
- Updates is a mapping between any UI component (at abstract or concrete level) and a domain attribute or instantiated attribute (at run time). Updates enable to specify that a UI component provides a value for the related domain concept.

Appendix B. UsiXML Ontology for User Interface Specification

- Triggers indicates a connection between a method of the domain model and a UI individual component (either at the abstract or at the concrete level).

Some other mappings are related to assure the transformations in order to achieve multi-path development of user interfaces. *Traceability mappings* are helpful for keeping a trace of the execution of the transformations. For instance it may be interesting to know which concrete object reifies which abstract object, or vice versa, which abstract object is an abstraction of which concrete object.

- Is Executed In maps a task to one or several AUI or CUI elements.
- Is Reified By indicates that a concrete object is the reification of an abstract one through a reification transformation.
- Is Abstracted Into indicates that an abstract object is the reification of a concrete one through an abstraction transformation.
- Is translated Into enables to provide a trace of the adaptation of one component in another, the transformation called translation.

Other useful mappings are:

- Manipulates maps a task to a domain concept. It may be an attribute, a set of attributes, a class (or an object), or a set of classes (or a set of objects). This relationship is useful when it comes to find the most appropriate interaction object to support a specific task.
- Has Context maps any model element to one or several contexts of use.
- IsShapedFor allows associating a plasticity domain to a CUI.

Appendix C. Three-Dimensional widgets representation

This section is based on examples developed in the Toolkits reviewed in the state of the art, and the examples shown in the case study, so as some shown inside the taxonomy. The use of the taxonomy provides hints to actual solutions for designers. More important is the fact that there can be more than the examples shown in each level, further investigation can fill the gaps or add more examples to the existing levels.

In addition, the possible renderings are examined. The comparison between the different representations of the same component that are presented were examined using criteria like the 2D and 3D consistency, the development complexity, the usability and how much intuitive the representation was. For this survey no formal evaluation was conducted but the survey was based on authors' experience [Kakl08a, Kakl08b].

The result of this survey pointed out that the best representation for a component depends on the special needs of the users. For instance, when the target group is the blind users, there is no need to keep a 2D-3D consistency, as users can't see the 2D components to make the comparison and more easily identify its 3D counterpart. In this case, the major criterion is to represent a 3D component in a way that is easy to comprehend and be used by the blind users, even if this representation has nothing to do with the 2D representation of the corresponding 2D components.

For the representation of the widgets the criteria proposed in [MacI91] were used. Accordingly to this method, a set of criteria can be evaluated when a question is raised and a set of possible answers exists (3D renderings in this case). In this case four criteria were identified to consider: 2D to 3D consistency, easy to develop, intuitional and the usability. The weight attributed to each criterion was based on the authors' past experience though the development of 3D and haptic applications.

Appendix C. Three-Dimensional widgets representation

The meaning of the links are: the darkest solid line (++) means strongly supported, dark solid line (+) means supported, solid line (~) means neutral, dash lines (-) means denied and dot lines (..) means strongly denied. Finally, the usability of any of these presentations would not be a problem; however is considered that the haptic and the 2D presentation could be more effective as they are more intuitional and have more consistency with the 2D already known component. Notice that this analysis only provides a general view of the different properties related to the development of the 3D widgets and does not arrives at a conclusion about which representation is generally the best. After presenting the graphical representation we present in two ways the evaluation criteria and the results, on a table and in a graphic.

1. Three-D Toggle Button

Definition: enables a Boolean choice by pushing a multi states button [USIX07]

Abstract attributes:

defaultContent: the text *component* string that defines the text displayed on the button. (Label attribute Inherit in Studierstube)

surroundedColor: the Appearance attribute that define the colour and texture of the button around the button, see the *black* colour described in Contigra.

centerColor: the Appearance attribute that define the colour and texture of the button in the centred of the button.

secondaryColor: the Appearance attribute that define the colour and texture of the button when the button change the state.

defaultState: Indicates a default state for a `toggleButton := TRUE FALSE`. [USIX07]

Abstract events:

EVT_TBT_click : the user clicks on *the* button.

Appendix C. Three-Dimensional widgets representation

What will be the representation of a 3D ToggleButton?	Switch	-	(1) 2D-3D Consistency
		--	(3) Easy to develop
		+	(4) Intuitional
		~	(5) Usability
	Sphere	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		~	(4) Intuitional
		~	(5) Usability
	2D representation	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Haptic	+	(1) 2D-3D Consistency
+		(3) Easy to develop	
+		(4) Intuitional	
+		(5) Usability	

Table C-1 Three-D Toggle Button evaluation table

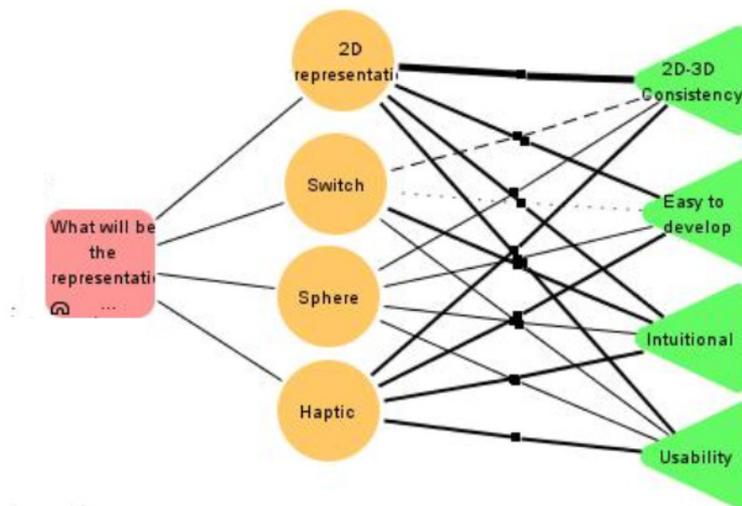


Figure C-1 Three-D Toggle Button evaluation diagram

Appendix C. Three-Dimensional widgets representation

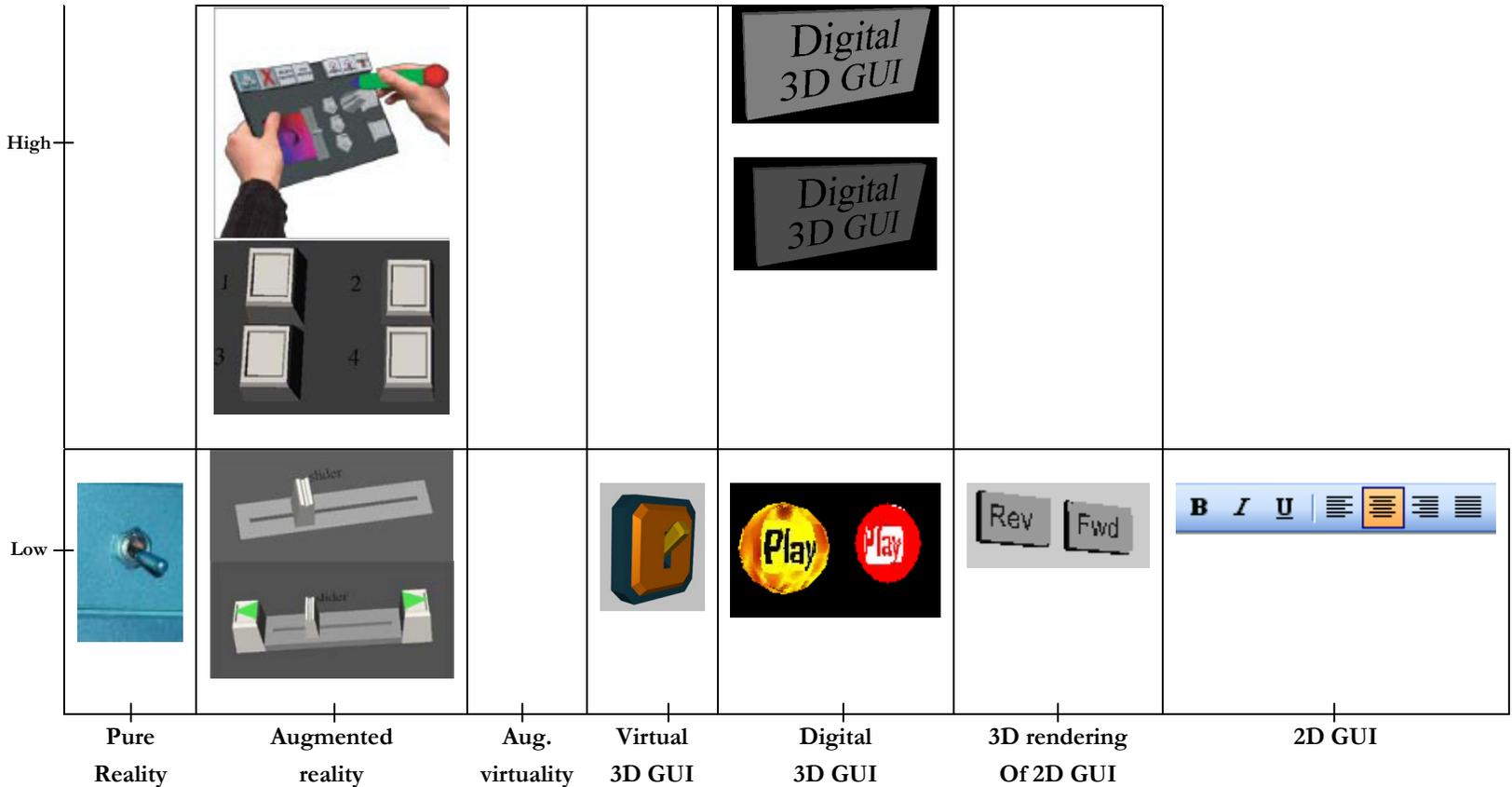


Figure C-2 Three-Dimensional Toggle Button Taxonomy

2. Three-D Text Component

Definition : textual sentence that is placed in different components of 3D words or by itself on the space. It is useful to explain the use of components, to define type or units of measure, to set titles.

Abstract attributes:

defaultContent : the vector of characters that compose the text of the text component. The text could be divided in different substrings and each one is place in one location of the vector, the purpose of doing this is to replace the use of the return character. [USIX07]

maxLength : define the size of the space where the text is displayed (the size is vertical or horizontal depending on the orientation of the text) := [0.0 (any length) ... ∞). [USIX07] (maxExtent in VRML)

maxLength : a vector that specifies the length of each substring of the text in the local coordinate system := [0.0 (any length) ... ∞). [USIX07] (Lengt in VRML)

textFont : define the family of the label (times, serif, ...). [USIX07] (FontFamily in VRML)

orientation : whether the text advance horizontally or vertically := horizontal, vertical.

advancing : define whether the text is written left to right or vice versa in the case of horizontal orientation, or top to bottom or vice versa in the case of vertical orientation := leftToRight, rightToLeft, bottomUp, topDown.

justification : determine alignment of the text := left, right and centre.

style : specifies the style of the text := PLAIN, BOLD, ITALIC, BOLDITALIC. (isBold is Italic in [USIX07])

textSize : specifies the high of the text := [0.0 (no size) ... ∞) [USIX07] (Size in VRML).

spacing : determine the line spacing between adjacent lines of text, this means, when more than one line of text compose the text displayed := [0.0 (no size) ...∞).

language : define the value of the language tag that is based on ISO 639:1988 := 'zh' for Chinese, 'jp' Japanese, 'sc' for Swedish.

defaultHyperlinkTarget : define a uri target := uri [USIX07].

Appendix C. Three-Dimensional widgets representation

What will be the representation of 3D Output Text?	2D representation	++	(1) 2D-3D Consistency
		~	(3) Easy to develop
		++	(4) Intuitional
		++	(5) Usability
	Haptic	--	(1) 2D-3D Consistency
		++	(3) Easy to develop
		--	(4) Intuitional
		+	(5) Usability

Table C-2 Three-D Output Text evaluation table

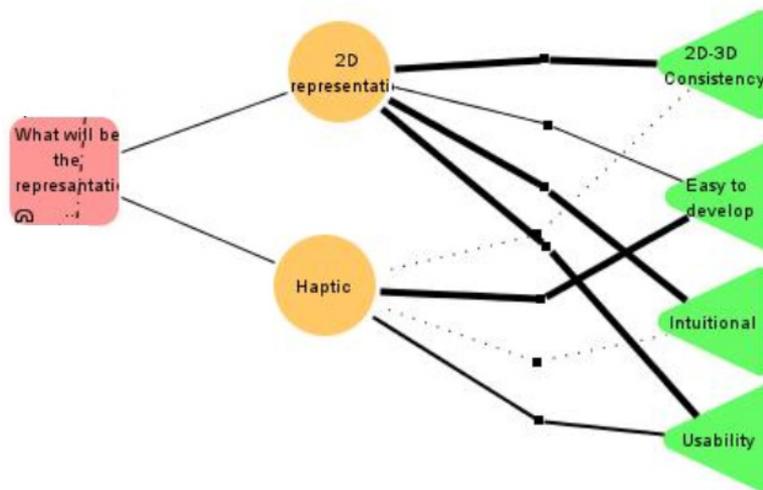


Figure C-3 Three-D Output Text evaluation diagram

Appendix C. Three-Dimensional widgets representation

What will be the representation of a 3D InputText?	3D text	++	(1) 2D-3D Consistency
		-	(3) Easy to develop
		++	(4) Intuitional
		++	(5) Usability
	2D representation	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		++	(4) Intuitional
		++	(5) Usability
	Haptic	--	(1) 2D-3D Consistency
		++	(3) Easy to develop
		--	(4) Intuitional
		~	(5) Usability

Table C-3 Three-D Input Text evaluation table

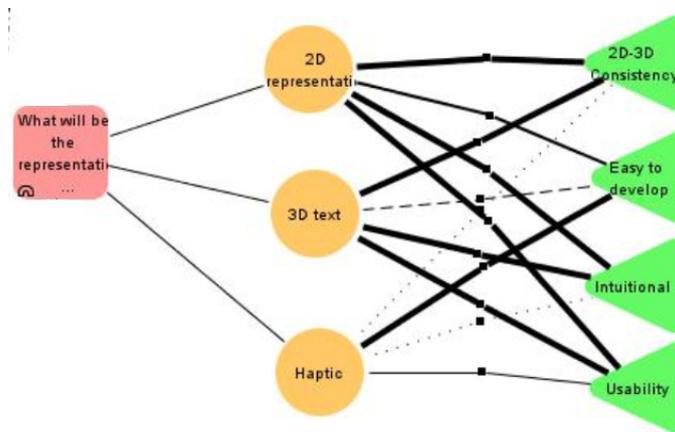


Figure C-4 Three-D Input Text evaluation diagram

Appendix C. Three-Dimensional widgets representation

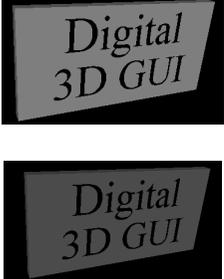
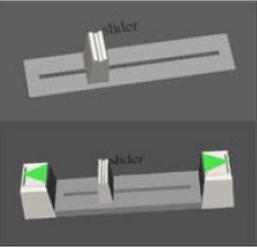
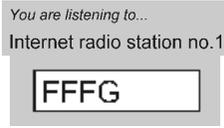
High							
Low							
	Pure Reality	Augmented Reality	Aug. virtuality	Virtual 3D GUI	Digital 3D GUI	3D rendering Of 2D GUI	2D GUI

Figure C-5 Three-Dimensional Text Component Taxonomy

Appendix C. Three-Dimensional widgets representation

3. Three-D Colour Picker

Definition: Enables to choose a colour within a palette. [USIX07].

Abstract attributes:

colour: a 3D colour selected := <red> <green> <blue> the three values are double.

What will be the representation of a 3D Colour Picker?	Traditional way	++	(1) 2D-3D Consistency
		--	(3) Easy to develop
		++	(4) Intuitional
		+	(5) Usability
	Innovative representation	~	(1) 2D-3D Consistency
		-	(3) Easy to develop
		~	(4) Intuitional
		~	(5) Usability

Table C-4 Three-D Colour Picker evaluation table

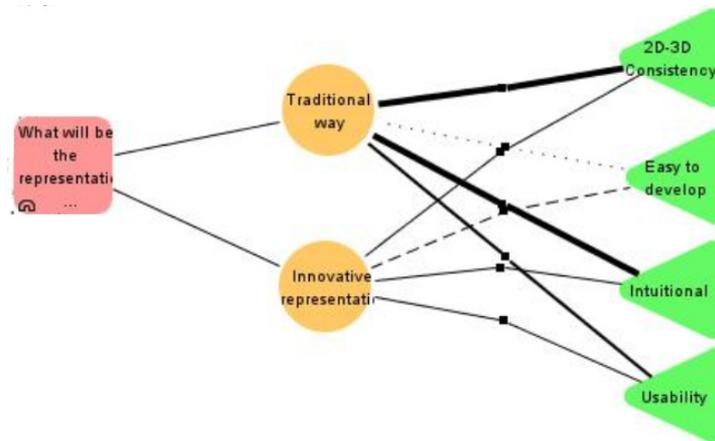


Figure C-6 Three-D Colour Picker evaluation diagram

Appendix C. Three-Dimensional widgets representation

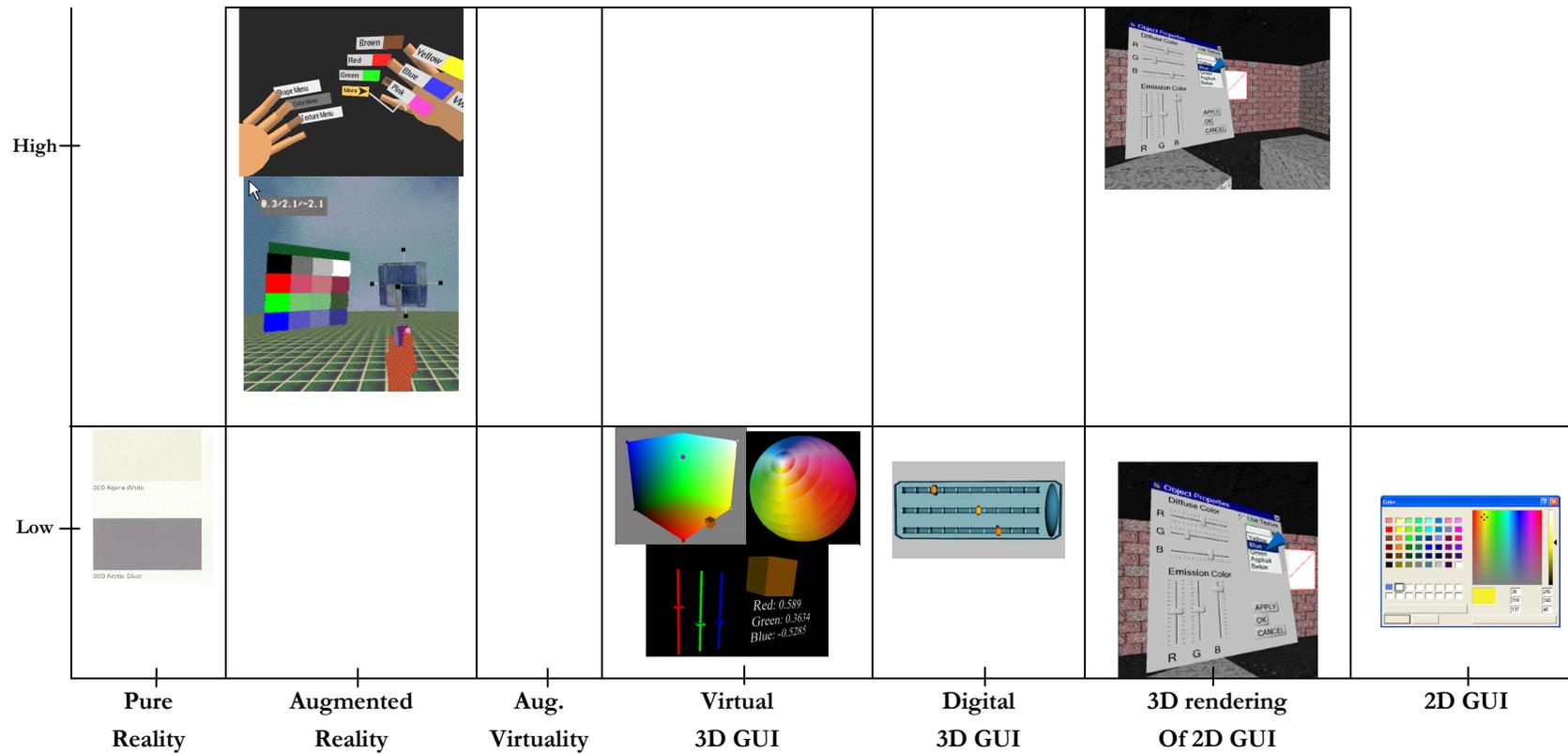


Figure C-7 Three-Dimensional Colour Picker

Appendix C. Three-Dimensional widgets representation

4. Three-D Radio Button

Definition: Enables a Boolean choice by checking a circle aside of a label. An optionButton may be differentiated from a checkBox by the fact that when grouped optionButton selection is mutually exclusive while checkBox allows multiple choices. [USIX07]

Abstract attributes:

defaultState: Indicates a default state for a toggleButton := TRUE FALSE [USIX07]

groupName: Is the name of the := String [USIX07]

What will be the representation of a 3D RadioButton?	3D representation	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Innovative representation	--	(1) 2D-3D Consistency
		-	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Haptic	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability

Table C-5 Three-D Radio Button evaluation table

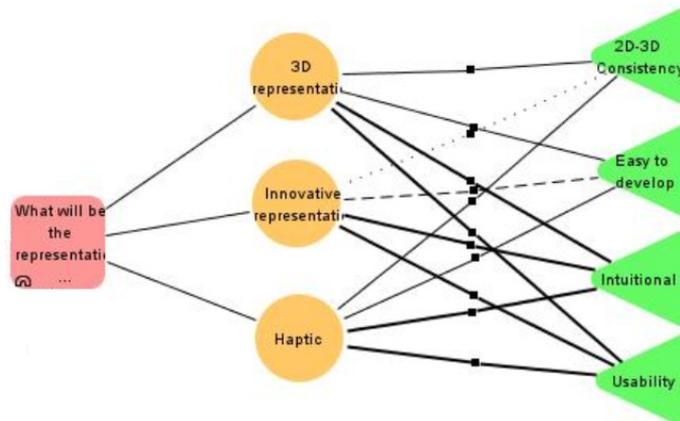


Figure C-8 Three-D Radio Button evaluation diagram

Appendix C. Three-Dimensional widgets representation

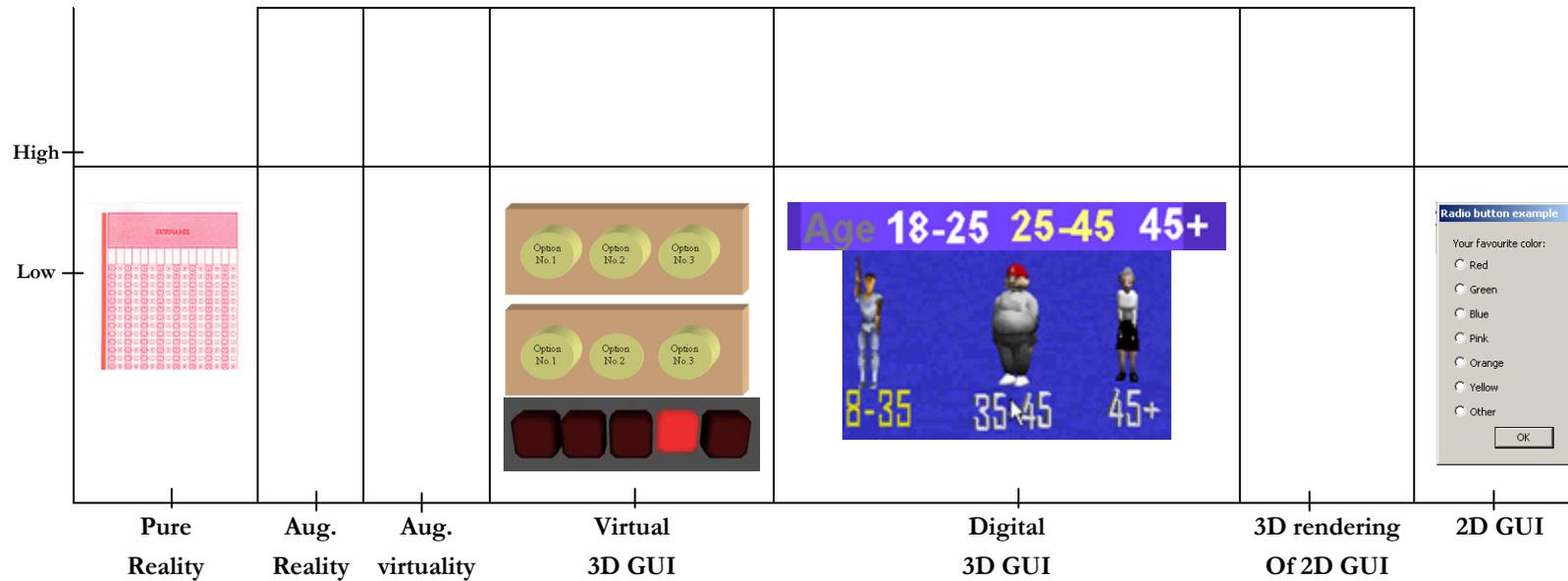


Figure C-9 Three-Dimensional Radio Button Taxonomy

Appendix C. Three-Dimensional widgets representation

5. Three-D Check Box

Definition: Enables a boolean choice by checking a square box aside of a label. A checkBox may be differentiated from a radio button (optionButton) by the fact that when grouped checkBoxes allow multiple choices while (optionButton) selection is mutually exclusive. [USIX07].

Abstract attributes:

defaultState: Indicates a default state for a check box := TRUE FALSE [USIX07]

groupName: Is the name of the group := String [USIX07].

What will be the representation of a 3D CheckBox?	2D Representation	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Haptic	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		-	(4) Intuitional
		+	(5) Usability

Table C-6 Three-D Check Box evaluation table

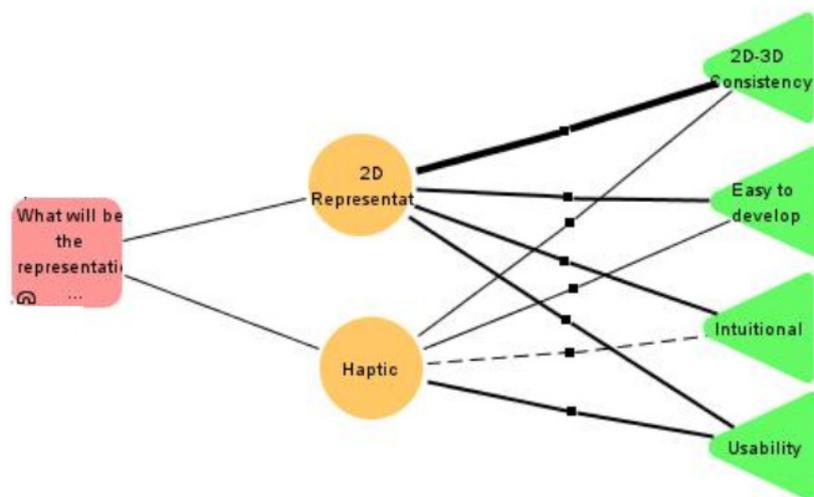


Figure C-10 Three-D Check Box evaluation diagram

Appendix C. Three-Dimensional widgets representation

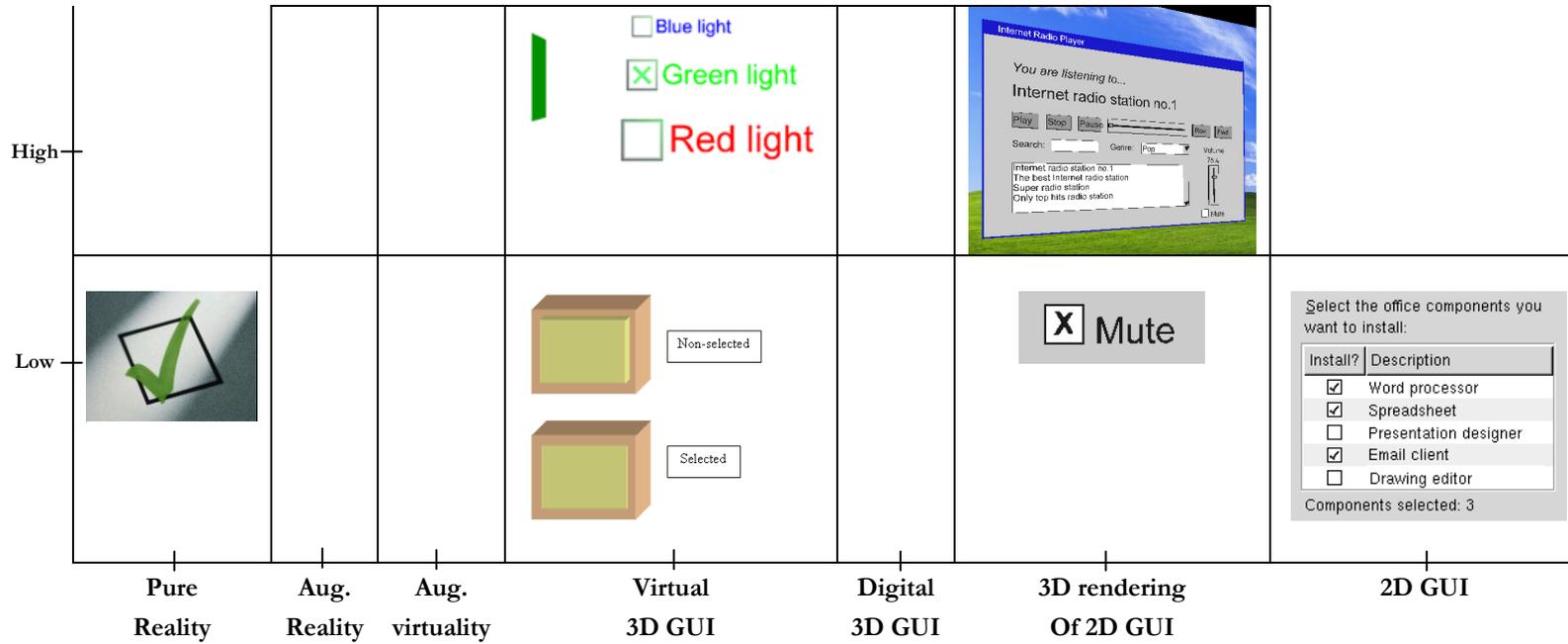


Figure C-11 Three-Dimensional Check Box Taxonomy

Appendix C. Three-Dimensional widgets representation

6. Three-D ComboBox

Definition: Enables a direct selection over a collection of sequentially ordered items. [USIX07]

Abstract attributes:

isEditable: Specifies if the content of the textbox (composing a comboBox) is editable or not:= TRUE or FALSE [USIX07]

maxLineVisible: Indicates the number of visible lines := integer [USIX07]

items: Indicate de content of the combo Box := item object [USIX07].

What will be the representation of a 3D ComboBox?	2D Representation	++	(1) 2D-3D Consistency
		--	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Haptic	--	(1) 2D-3D Consistency
		+	(3) Easy to develop
		-	(4) Intuitional
		+	(5) Usability

Table C-7 Three-D Combo Box evaluation table

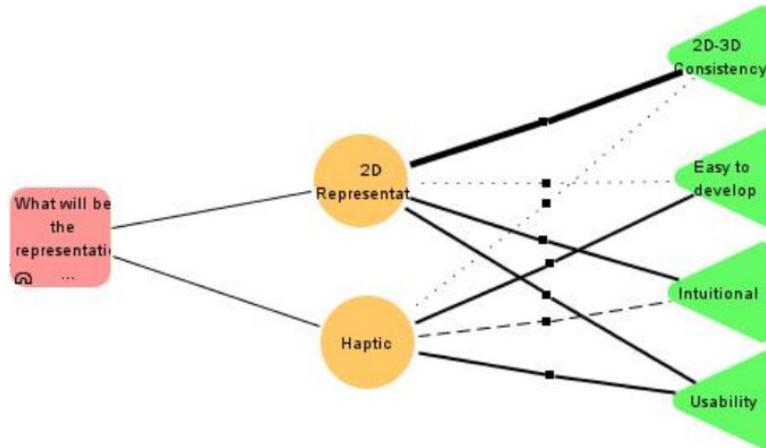


Figure C-12 Three-D Combo Box evaluation diagram

Appendix C. Three-Dimensional widgets representation

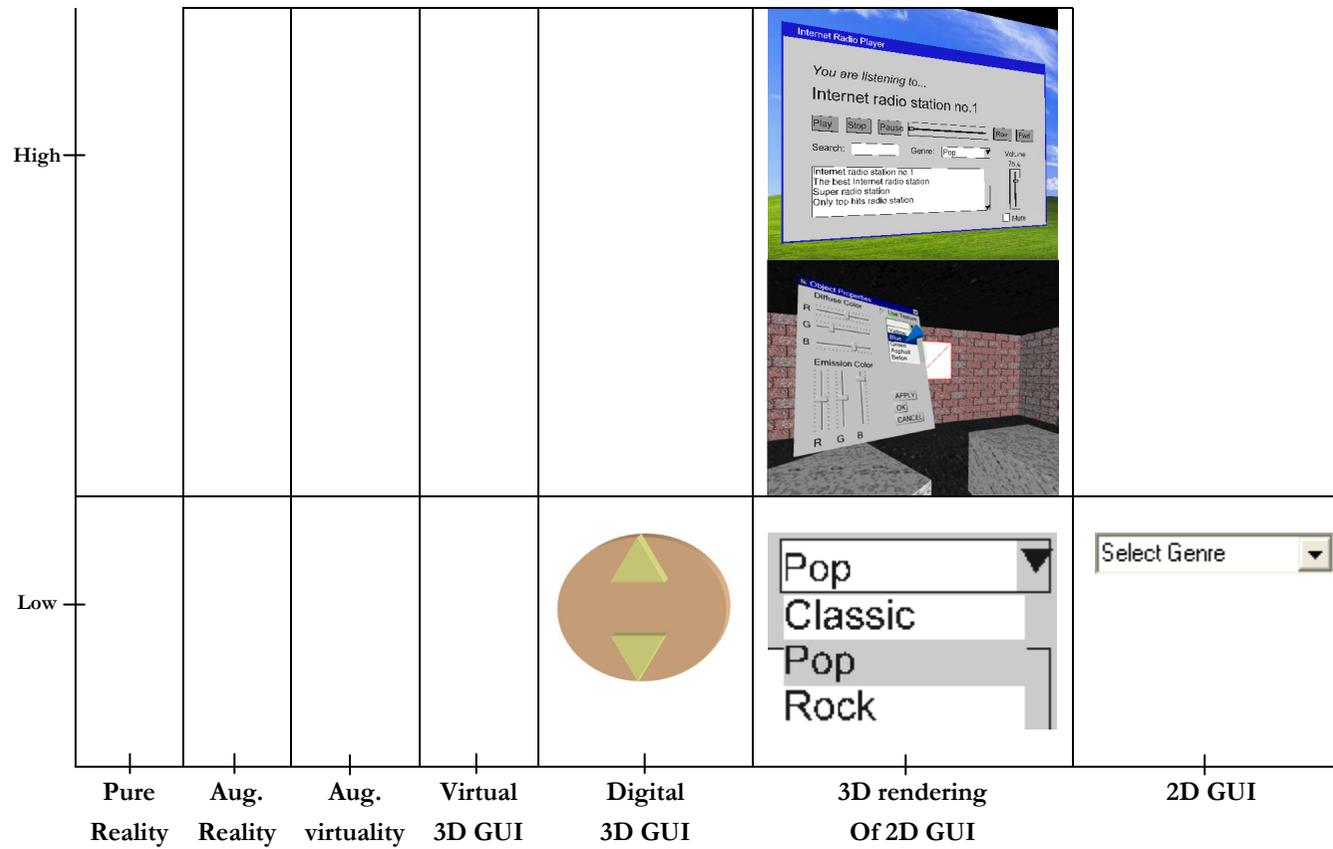


Figure C-13 Three-Dimensional ComboBox Taxonomy

7. Three-D List Box

Definition: Enables a selection over a single or multiple selection drop-down list. [USIX07]

Abstract attributes:

isEditable: Specifies if the content of the listBox is editable or not:= TRUE or FALSE [USIX07]

maxLineVisible: Indicates the number of visible lines := integer [USIX07]

items: Indicate de content of the list Box := item object [USIX07].

We did not evaluate the List Box.

Appendix C. Three-Dimensional widgets representation

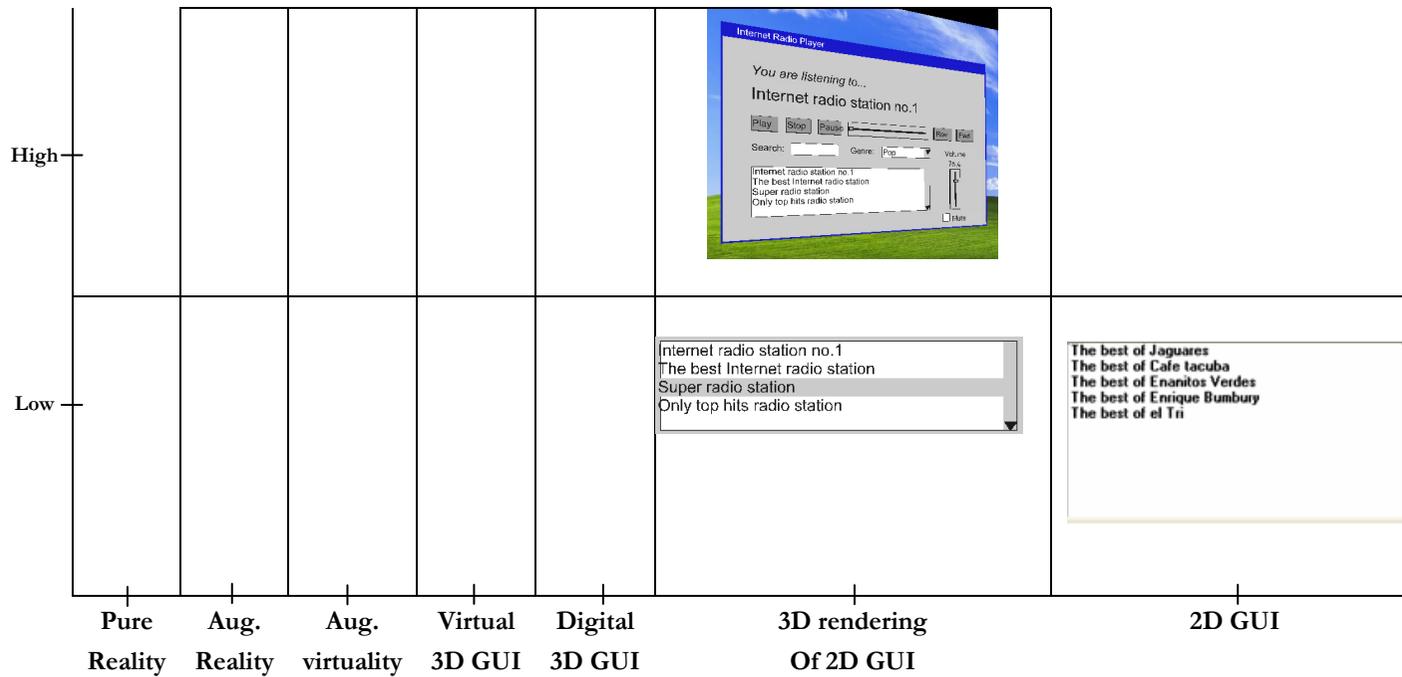


Figure C-14 Three-Dimensional ListBox Taxonomy

Appendix C. Three-Dimensional widgets representation

8. Three-D Button

Definition : Allows a selection of one or two integer value(s) over a set of ordered integers [USIX07]

Abstract attributes:

minValue : Is the lower bound of slider values := integer [USIX07] idem Studierstube

maxValue : Is the upper bound of slider value := integer [USIX07] idem Studierstube

step : is to precise intermediate slider values between minValue and maxValue := integer [USIX07]

orientation : define the orientation of the slider := string equals vertical or horizontal [USIX07]

cursorPosition : Indicates a default value for a cursor, that allows to choose a value on a slider := integer [USIX07]

defaultContent : the text component string that defines the text displayed on the slider. [USIX07]

increment : define how much the increment or decrement buttons will move the slider := double value (Attribute in Studierstube).

What will be the representation of a 3D button?	Traditional way	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Sphere	-	(1) 2D-3D Consistency
		+	(3) Easy to develop
		-	(4) Intuitional
		~	(5) Usability
	Haptic	+	(1) 2D-3D Consistency
		+	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability

Table C-8 Three-D Button evaluation table

Appendix C. Three-Dimensional widgets representation

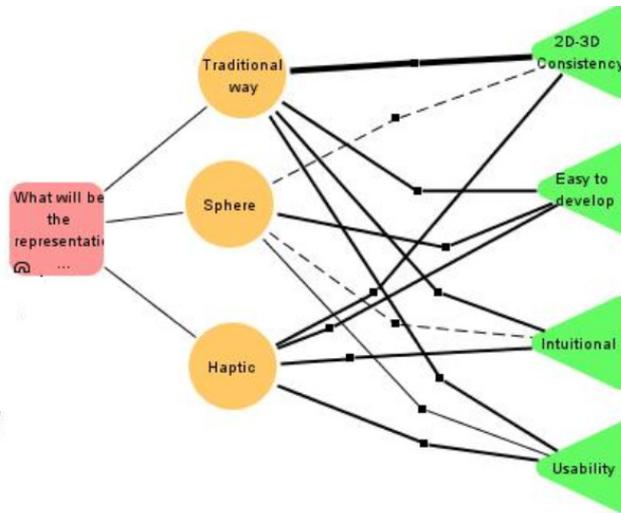


Figure C-15 Three-D Button evaluation diagram

Appendix C. Three-Dimensional widgets representation

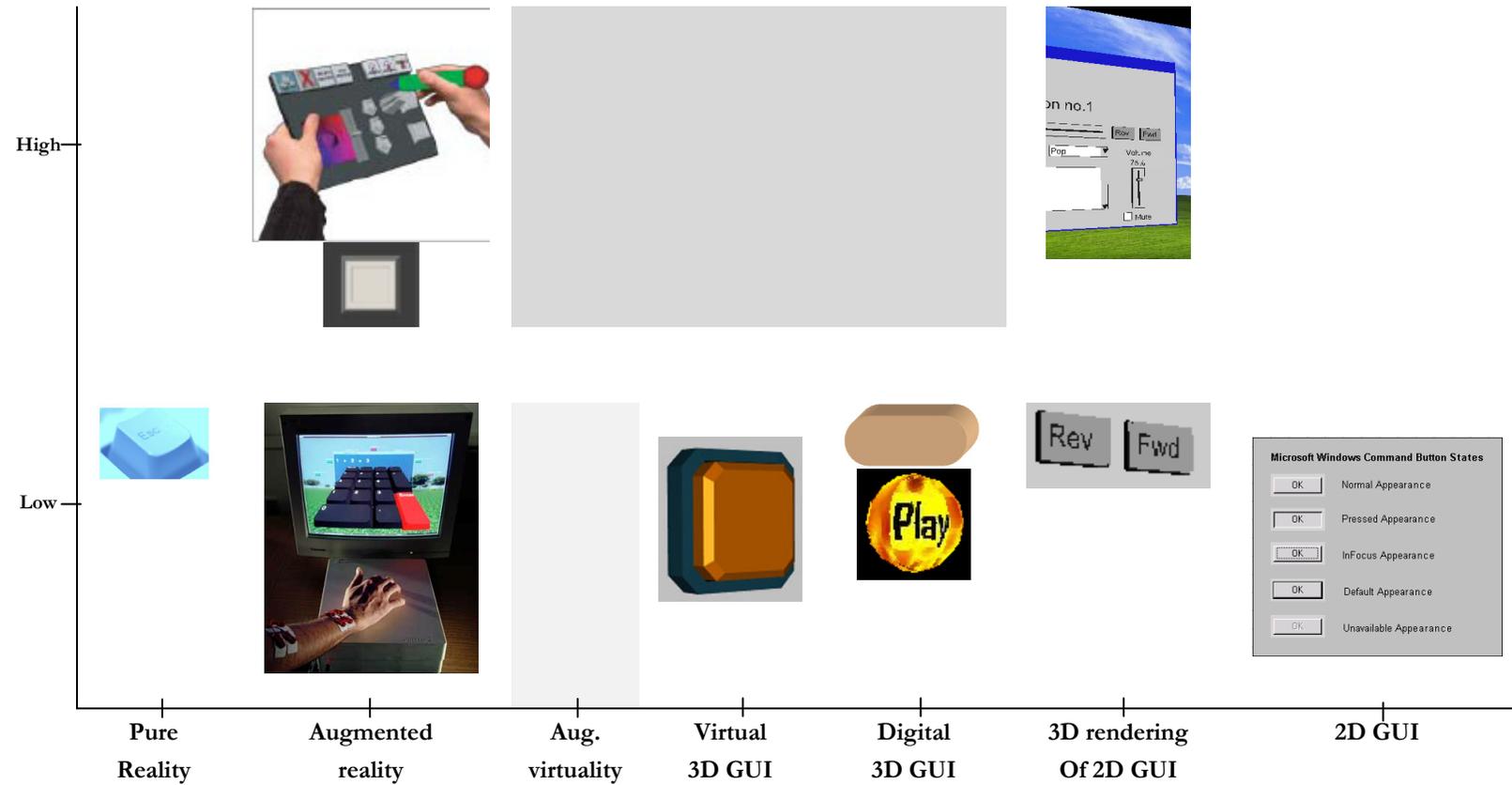


Figure C-16 Three-Dimensional Button Taxonomy

9. Hyperlink

Navigation button



Haptic



Special for documents



What will be the representation of a 3D Hyperlink?	Navigation button	+	(1) 2D-3D Consistency
		+	(3) Easy to develop
		++	(4) Intuitional
		+	(5) Usability
	Haptic	~	(1) 2D-3D Consistency
		~	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability

Table C-9 Three-D Hyperlink evaluation table

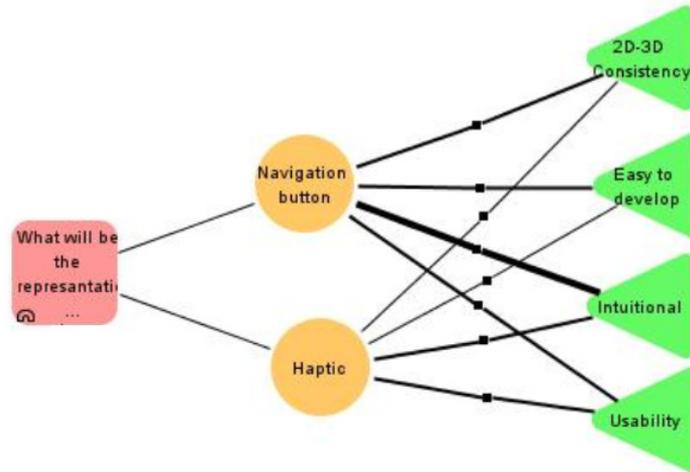
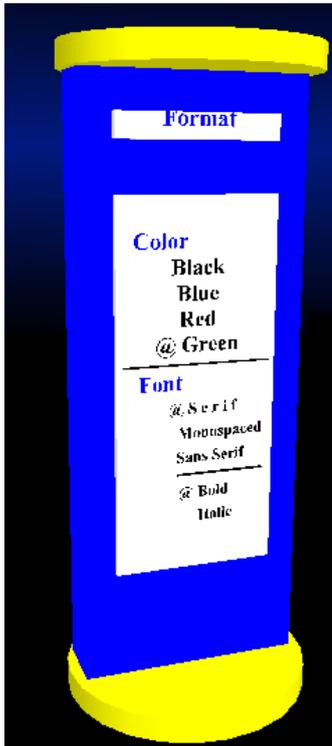


Figure C-17 Three-D Hyperlink evaluation diagram

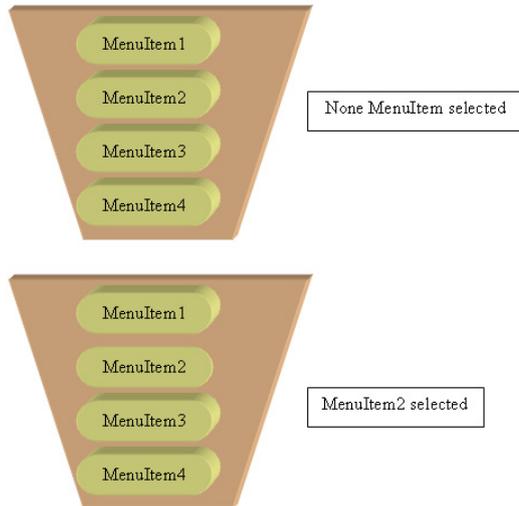
10. Menu and Menu item

Innovative



Appendix C. Three-Dimensional widgets representation

Haptic



What will be the representation of a 3D Menu and its 3D MenuItems?	Innovative representation	-	(1) 2D-3D Consistency
		--	(3) Easy to develop
		~	(4) Intuitional
		-	(5) Usability
	Haptic	-	(1) 2D-3D Consistency
		~	(3) Easy to develop
		-	(4) Intuitional
		~	(5) Usability

Table C-10 Three-D Menu evaluation table

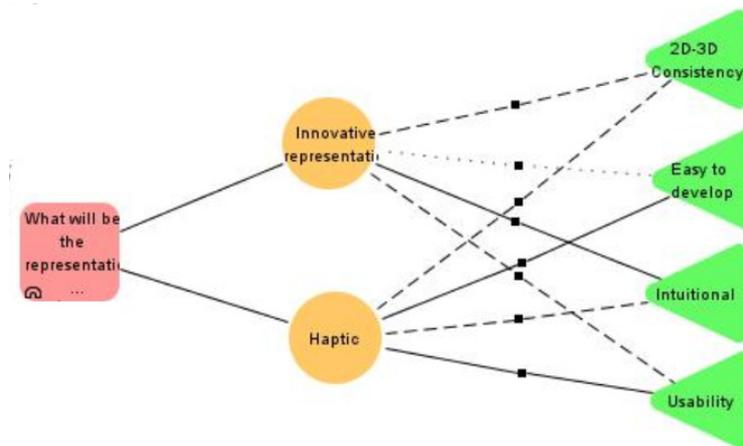
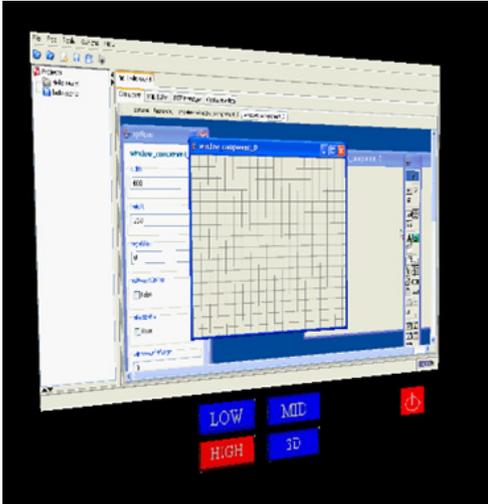


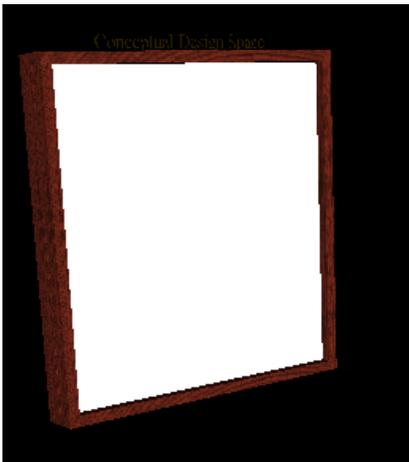
Figure C-18 Three-D Menu evaluation diagram

11. Image Component

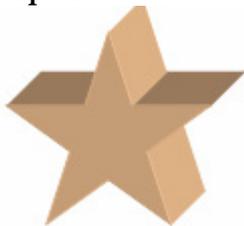
Traditional



Innovative



Haptic



Appendix C. Three-Dimensional widgets representation

What will be the representation of a 3D Image?	Traditional	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		++	(4) Intuitional
		++	(5) Usability
	Innovative	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		++	(5) Usability
		++	(4) Intuitional
	Haptic	--	(1) 2D-3D Consistency
		+	(3) Easy to develop
		-	(4) Intuitional
		~	(5) Usability

Table C-11 Three-D Image Component evaluation table

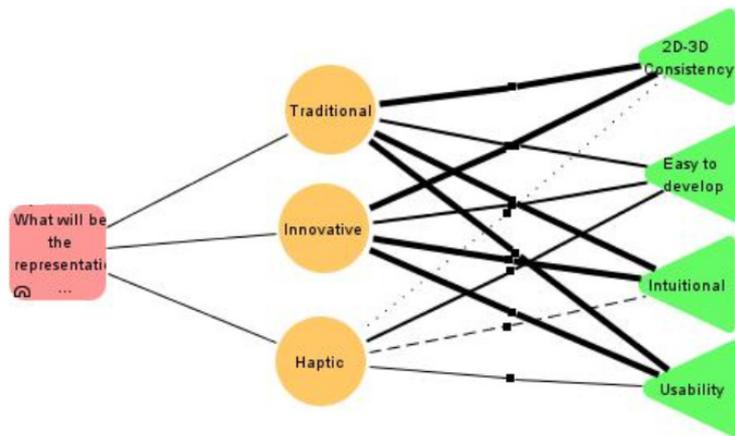
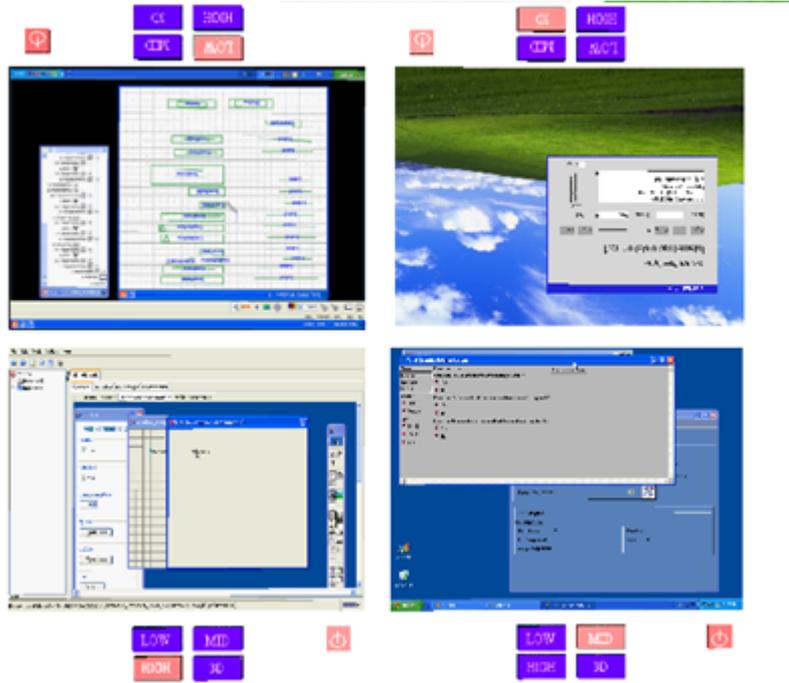


Figure C-19 Three-D Image Component evaluation diagram

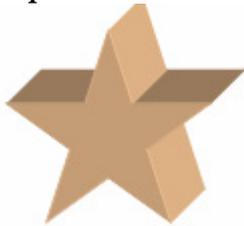
Appendix C. Three-Dimensional widgets representation

12. Video Component

Traditional



Haptic



Appendix C. Three-Dimensional widgets representation

What will be the representation of a 3D Video?	Traditional	++	(1) 2D-3D Consistency
		+	(3) Easy to develop
		++	(4) Intuitional
		++	(5) Usability
	Haptic	--	(1) 2D-3D Consistency
		++	(3) Easy to develop
		-	(4) Intuitional
		~	(5) Usability

Table C-12 Three-D Video Component evaluation table

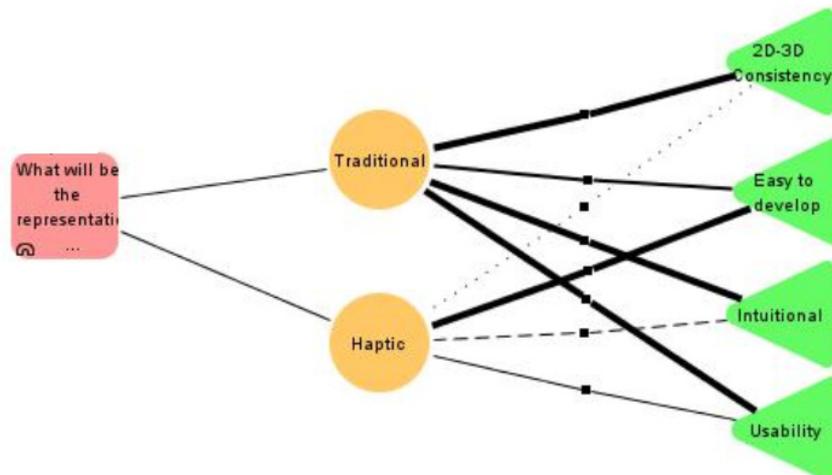


Figure C-20 Three-D Video Component

Appendix D. Canonical List of Task Types

In the appendix we describe in detail each category of the task types proposed, providing some examples of the results of mapping different *user categories*(user, system and interactive) and *task items* to each *task type*. This is still an ongoing work.

1. Communicate

Communicate refers to the action of exchanging information between the user and the system. This communication channel uses several means to achieve its goal. Depending on the modality and the available devices we can show the results of a search in a table. Some other related actions that involve the communication channel are: transmit, call, acknowledge, respond/answer, suggest, direct, instruct, request.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Communicate		A search button	Record information in a database	Acknowledge the information Receive a communication
		Acknowledge the results of a search [a search in Google]	Communicate the results of a search	Transmit a communication
		Ask a question	Answer to a question Show the status bar	
		Request information function	Provide information function	

Table D-1 Communicate User Interface actions examples

Appendix D. Canonical List of Task Types

2. Create

Creating items has a variety of sub types and synonyms. It could mean: associate items, input/Encode/Enter, group, introduce, insert, new, assemble, aggregate, overlay (cover), and add. All of them imply the creation of a new item, or an instance of an item.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Create		New customer	A robot system that create a new customer	To have the intention of any sort of creation
		Blank slide	A robot system that create a new blank slide	
		New category	A robot system that create a new category	
		Create a new combination of CTRL keys	Insert a registry in a database	

Table D-2 Create User Interface actions examples

Appendix D. Canonical List of Task Types

3. Delete

The action of deleting an item corresponds to the opposite to creating an item, i.e. Remove/cut, ungroup, disassociate.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Delete		Remove a customer	After a period of inactivity remove an email account	To have the intention of any sort of deletion
		Delete a slide	Remove help windows after x time of not using it.	
		Break network connection	After a period of inactivity remove an icon of the desktop	
		Remove automatic download of e-mails function	After closing the MSN several services are close automatically, such as the data transfer between users.	

Table D-3 Delete User Interface actions examples

Appendix D. Canonical List of Task Types

4. Duplicate

Duplicating specifies the copy of an item. This task type has also some synonyms such as: clone, twin, reproduce, copy, and instance.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Duplicate		Copy user info	Copy user information after login a system	To have the intention of any sort of duplication
		duplicate slide	Copy slide format when inserting a new one	
		copy address	Copy user name	
		Duplicate animation on an object	While installing a software copy the control files to launch the installation.	

Table D-4 Duplicate User Interface actions examples

Appendix D. Canonical List of Task Types

5. Filter

The action of filtering a task item. A common example is the filter of email applications.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Filter		Filter email by groups	Automatically filter data accordingly to predefined values	Make the decision of filtering
		Filter results of a search	Filter the UI accordingly to the context of use	
		Filter an email as junk	Automatically filter an email as junk	
		Filter applications with know fabricants	Filter an application while trying to execute	

Table D-5 Filter User Interface actions examples

Appendix D. Canonical List of Task Types

6. Mediate

Mediate refers principally to the cognitive process of the user before making a decision. The system also could have some means to help the user in his cognitive process. So there is no interaction at this level. The user is in the process of: analyze, synthesize, compare, evaluate, decide, interpolate, integrate, formulate, among others.

A previous task must provide means to help the user accomplish his task, for instance, by showing combo boxes, radio button groups with available options.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Mediate		Compare products by price	Google search <i>evaluating</i> the best ranked pages to present the results of a query.	Analyze the data details (author, name, publisher, ...) of a book
		Compare side by side documents in word	Decide the layout of a slide when creating a new one	Compare a list of books
		Evaluate a video watched on youtube	Evaluate the security risk of a password	Determine the date of a trip
		Decide which operation to apply to a combination of CTRL keys.	Propose different arrangement of the results of a query.	Decide which operation will be used with a special key on a joystick

Table D-6 Mediate User Interface actions examples

Appendix D. Canonical List of Task Types

7. Modify

The modification of an item has several related subtypes: alter, transform, turning, rename, segregate, resize.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Modify		Edit client details	Resize a composed object	The intention of modifying an item
		Collapse a tree view	Change the layout of the UI accordingly to the context	
		Change shipping address	Turning automatically the volume	
		Change a combination of CTRL keys	Update automatically the date/time of the computer	

Table D-7 Modify User Interface actions examples

Appendix D. Canonical List of Task Types

8. Move

The action to change the location of a task item.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Move		Move user information to other page or system	Reallocate a table in a document because of the space	The intention of moving an item
		Move a window	Reallocate a tool bar because of the space	
		To Put email address into address list	Move automatically a n email address to other line	
		Reallocate a <i>Fx</i> from one cell to another cell in a Excel page		

Table D-8 Move User Interface actions examples

Appendix D. Canonical List of Task Types

9. Navigation

The action to find the way through containers

Task Type	Task Item	User category		
		Interactive 	System 	User 
Navigation		Pass from one item to another in a form	Passing one by one a register in a data base	The intention of navigate
		Going from one slide to other slide	Navigate among Word pages to find a specific word	
		Navigation inside a combo box with the navigation bar	Pass one by one a file in a search	
		Navigation between the date/time functions	Navigation among the configuration functions at the turn on moment	

Table D-9 Navigation User Interface actions examples

Appendix D. Canonical List of Task Types

10. Perceive

Acquire/detect/search for/scan/extract, identify / discriminate / recognize, Locate, Examine, monitor, scan, detect, the action of identifying items and/or information from the items.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Perceive		-----	Command recognition in a speech recognition system	Sound emitted by the system
		-----	Extract container structure	Observe windows transitions (fade-in, fade-out)
		-----	Recognize	Locate a destination on a map
		-----	Detect	---

Table D-10 Perceive User Interface actions examples

Appendix D. Canonical List of Task Types

11. Reinitialize

The action of reinitializing an item which in a UI imply some other modifications such as erase or clean certain fields (text field in the graphical modality). At the data level it implies to restore the default value. In some cases the task type is implicit in some other task types, for instance, when creating a new account there are fields to fill, independently on the modality, it is always possible to erase an entry. This do not means that for each entry there will be a need for a supplementary task to specify that it can be reinitialized. This is something that at the implementation level is assumed. On the contrary, when we reinitialize a *collection* this has a dipper and more general impact.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Reinitialize		All the customer registration elements (name, address) are set to their default values	A system response to restore an item to its default value.	Make the decision of reinitializing a task item
		A button that clear a form or restore to the default values		
		Erasing a text field.		
		Pressing a button to restore a variable to its default value		

Table D-11 Reinitialize User Interface actions examples

Appendix D. Canonical List of Task Types

12. Select

Choose, Pick selection between items group member picker, object selector.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Select		Select some fields of a form	Select some fields of a data base to fill	Have the intention of selecting
		Select a windows to copy	Choose an application	
		Choose an email address from a list	Pick a field of a data base	
		Pick a CTRL key	Select automatically the date/time function to update	

Table D-12 Select User Interface actions examples

Appendix D. Canonical List of Task Types

13. Start

Play, Search, active, execute, function, purchase, record, Specifies the beginning of an operation Play audio/video file initiate, launch??

Task Type	Task Item	User category		
		Interactive 	System 	User 
Start		Start to fill out a form	Starting to save information in a data base	The intention of starting
		Start a power point presentation	Active an application	
		Play a video file	Active a field in an application	
		Execute the virus scan	Execute automatically the virus scan	

Table D-13 Start User Interface actions examples

Appendix D. Canonical List of Task Types

14. Stop

End / finish/exit/suspend?/complete? Specifies the end of an action Stop searching/playing, cancel register.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Stop		Finish to fill out a form	Exit of a data base	The intention of stopping
		Exit of an application	After a period of inactivity close an application	
		Stop a video file	Cancel register	
		Suspend the virus scan	Complete the virus scan	

Table D-14 Stop User Interface actions examples

Appendix D. Canonical List of Task Types

15. Toggle

Activate/ deactivate, The existence of two different states of an item Bold on/off, encrypted mode.

Task Type	Task Item	User category		
		Interactive 	System 	User 
Toggle		Activate / deactivate a group of radio buttons	Activate / deactivate automatically a form	Make the decision of toggle an option
		Switch among Micro-soft office applications	Switch to another container as soon the previous one is finish. For instance in form when some data is fill out the cursor change automatically to another container with more fields to fill	
		Activate / deactivate a check box	Activate / deactivate a command in a menu	
		Activate / deactivate virus scan	Activate / deactivate message power battery	

Table D-15 Toggle User Interface actions examples

Appendix E. Three Dimensional User Interfaces Taxonomy

Computational models are used in a wide range of application in exact sciences, so as in social and economical sciences [Robe83]. Two tasks are required to construct models. Firstly, the conceptual aspect that involves: problem definition, system conceptualization and representation of the model. Second, the technical aspect that involves: the behaviour of the model, the evaluation and the policy of analysis and use of the model. To generate 3DUI models we have to evaluate and analyze 3DUI in the literature to extract the relevant characteristics of present developments.

For this purpose, taxonomy the applications will be necessary. This will be useful to identify the context of use of such applications. Also, will help us to understand the task in a more detailed level and how certain techniques address the task [Bowm00]. With taxonomies and implemented tools, hybrid models could be created or tested. New possibilities could be explored.

In our research we are proposing a new taxonomy that extends Milgram and Kishino continuum of mixed reality, [Milg94]. We consider that there is a need of a wider spectrum for virtual reality applications and not just mixed reality. In this way we can identify different basic or advanced mechanisms and techniques for virtualizing a user interface. We add a more continuous range of UIs in the virtual part since it can be a 2D UI, a 3D rendering of a 2D UI, a genuine 3DUI manipulating 3D objects, and so forth. Moreover, we reuse the degree of immersion that is allowed at each step: low and high immersion.

In the remaining of this appendix, in section 1 we present the exiting 3DUIs taxonomies. In section 2 the new taxonomy. Finally in section concludes this appendix.

1. Existing three-dimensional Taxonomies

Several taxonomies have been proposed for 3DUI based on different characteristics, such as: interaction techniques [Poup98], metaphors [Bowm00], and 3D

Appendix E. Three Dimensional User Interfaces Taxonomy

widgets [Dach02]. Each taxonomy fills their own requirements and as a result they are different one to each other. We review Milgram and Kishino continuum of mixed reality, [Milg94] in more detail as we consider it covers a wider spectrum of virtual applications.

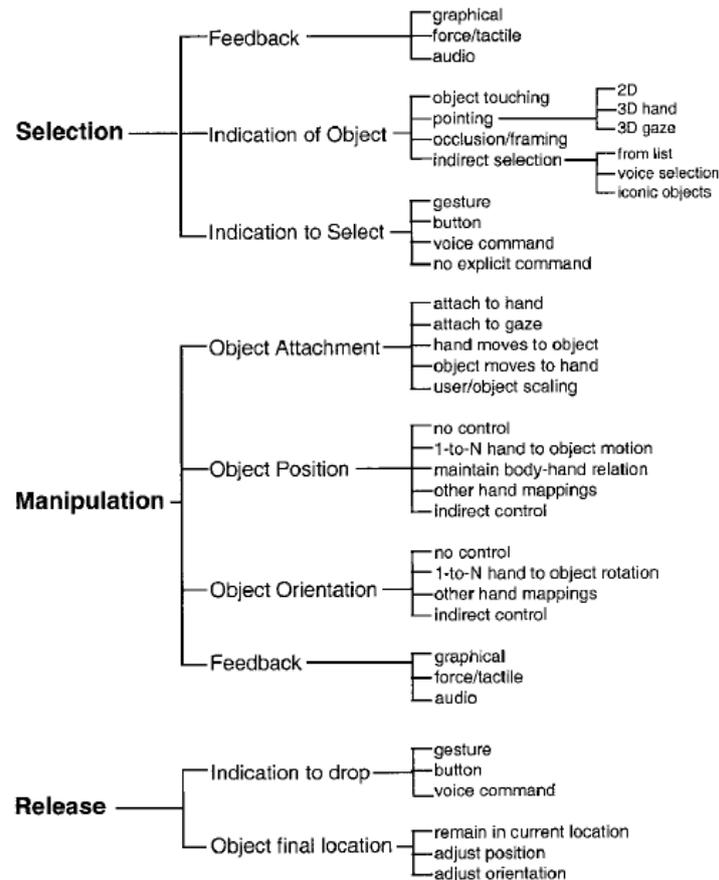


Figure E-1 Taxonomy of selection/manipulation techniques

A. Interaction Techniques Taxonomy

In [Bowm01a] four main 3D interaction tasks were identified for complex 3D applications, they are:

1. *Navigation* the most common VE task and is subdivided in two tasks, which are:
 - a. *Travel* refers to the physical movement from place to place.
 - b. *Wayfinding* the cognitive or decision-making component of navigation, and it asks the questions, “where am I?”, “where do I
2. *Selection* is the picking of an object or a set of objects for some purpose.

Appendix E. Three Dimensional User Interfaces Taxonomy

3. *Manipulation* refers to the specification of object properties, such as: *position* and *Orientation*.
4. *System Control* is the task of changing the system state or the mode of interaction. Examples in 2D that do this are menus or command-line interfaces.
 - a. *Implicit*
 - b. *Explicit*

In their review they analyzed universal task and create taxonomies of techniques. Based on the taxonomies they were able to anticipate performance of new interaction techniques. In Figure E-1 we show the taxonomy of selection/manipulation technique.

In this taxonomy the authors shows different approaches to some basic tasks, for instance, a 3D hand pointing to indicate an object for selection. As our goal is on the presentation more than the means to achieve tasks, this taxonomy will not be useful.

B. Metaphors Taxonomy

In a task model tree view [Pate97] we show in Figure E-2 the metaphors taxonomy for the manipulation task, inspired in [Poup98] metaphor taxonomy. The user has several means to achieve its manipulation task.

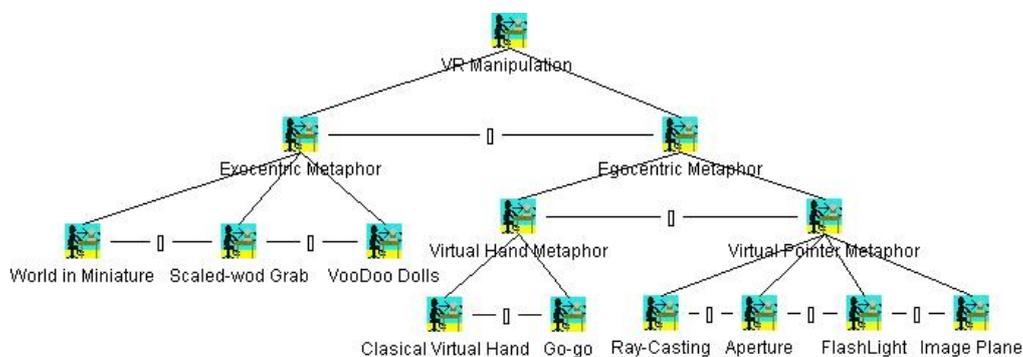


Figure E-2 Adaptation to the taxonomy of virtual metaphors [Poup98]

A metaphor in a VE is not an “atomic” but rather a “composed” interaction technique, which makes it harder to model. Leafs of the task model correspond to an Interaction technique itself, see previous section. The degree of immersion, characteristic that is depicted as exocentric/egocentric metaphors, is of our interest as we consider that it is important to differentiate between immersive applications and desktop based. We will come back to this later in this chapter.

Appendix E. Three Dimensional User Interfaces Taxonomy

C. 3D Widgets taxonomy

Another important Taxonomy to consider is Contigra [Dach02], whose objectives not only include the standardization of a repertoire of 3D widgets, but also metaphors and interaction techniques, everything structured in the form a hierarchy. Among those widgets, the repertoire includes some of the elements that are used in almost every 2D and 3D application, such as the button and the toggle button, but also other widgets that are only specific to 3D environments, such as the ring menu. The disadvantage is that many of the widgets of this hierarchy have not been developed or are not publicly available yet, which limits its adoption as a standard in the field. Also it is not clear how they were implemented or if there models for each widget.

Nevertheless this is the closest taxonomy that we found for our purposes. We need to identify which were the existing representations for widgets. So Contigra taxonomy helps us to identify much of such efforts. Even so, there were still a lack of separation of degree of immersion and the levels of virtuality, we will come back to this in next section.

The screenshot displays the Contigra Components interface. On the left is a hierarchical tree of 3D widgets, including categories like 3D Widgets, Direct 3D Object Interaction, 3D Scene Manipulation, Exploration and Visualization, System / Application Control, State Control / Discrete Valuators, Activation, Push Button, Menu Selection, Two States, Push Switch, Toggle Switch, Slide Switch, Check Box, Multiple States, Radio Button, Dial Switch, Multiple Check Boxes, Continuous Valuators, Scalar Values, Dial, Up-Down Switch, Slider, Scroll Bar, Multiple Values, PlaneSlider, Color Chooser, Special Value Input, Containers, and Menu Selection. The right pane shows the 'Toggle Switch Component' details, including a 3D preview, version (1.0), developer (Michael Hinz), and license (none). It features two tables: 'Geometry Parameters' and 'Behavior Parameters'.

Geometry Parameters	dataType	configurable	receivesEvents	generatesEvents		
KnobGeometry The geometry of the knob.	CoAnyURI	×			×	×
LabelText text that titles the button	CoStrings	×			×	
ShellGeometry This geometry frames the knob and is fixed.	CoGroup	×			×	

Behavior Parameters	dataType	configurable	receivesEvents	generatesEvents		
BooleanState boolean state of the PushSwitch (true = on, false = off)	CoBoolean	×	×	×	×	×

Michael Hinz
Dresden University of Technology
<http://www.contigra.com>

TU Dresden, Lehrstuhl Multimediatechnik
Dresden
Contigra@tu-dresden.de

This widget classification and component hierarchy is part of the CONTIGRA research project. © 2005 TU Dresden
To interactively view a component click on its image. We recommend the Cortona Plugin: [Download Cortona](#)
Best results are achieved with Internet Explorer. In case of problems with the behaviors, just install Microsoft Virtual Machine [Download \(5.21MB\)](#)

Figure E-3 Contigra widgets taxonomy

D. Mixed reality continuum Taxonomy

Milgram and Kishino continuum of mixed reality [Milg94] defines a continuum of real-to-virtual environments between Mixed Reality (MR). Their objective was

Appendix E. Three Dimensional User Interfaces Taxonomy

the concept of having both “virtual space” on the one hand and “reality” on the other available within the same visual display environment.

The concept of a “virtuality continuum” relates to the mixture of classes of objects presented in any particular display situation, as illustrated in Figure E-4, where real environments, are shown at one end of the continuum, and virtual environments, at the opposite extreme.

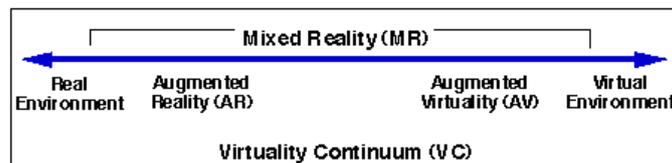


Figure E-4 Simplified representation of a "virtuality continuum"

Milgram and Kishino differentiate their continuum based on the devices that render the projection of the virtual world. The diversification of devices produce the different sensation of immersion, six displays were described: (1) monitor based (non-immersive) video displays, (2) the same video displays but using a head mounted display (HMD) rather than monitors, (3) HMD equipped with see through capability, (4) The same as 3 but using a video display of the real world, (5) completely graphic display environments, completely immersive, partially immersive or otherwise, to which video reality is added, and finally, (6) completely graphic but partially immersive environments.

Their taxonomy helps to distinguish among the various technological requirements necessary for realizing and researching MR displays, with no restrictions on whether the environment is supposedly immersive (HMD based) or not. To do so, they propose three questions that should be solved, See Figure E-5.

How much do we know about the world being displayed? The answer is what they called Extent of World Knowledge. On the left extreme no information is known about the world to be displayed, on the other extreme, complete information is known. In the middle of the two poles there are three scenarios, the system will identify where are the objects but not what are they, second the system knows what are the objects but no where they are, and the last one, the system knows both where the objects are and what are they.

Appendix E. Three Dimensional User Interfaces Taxonomy

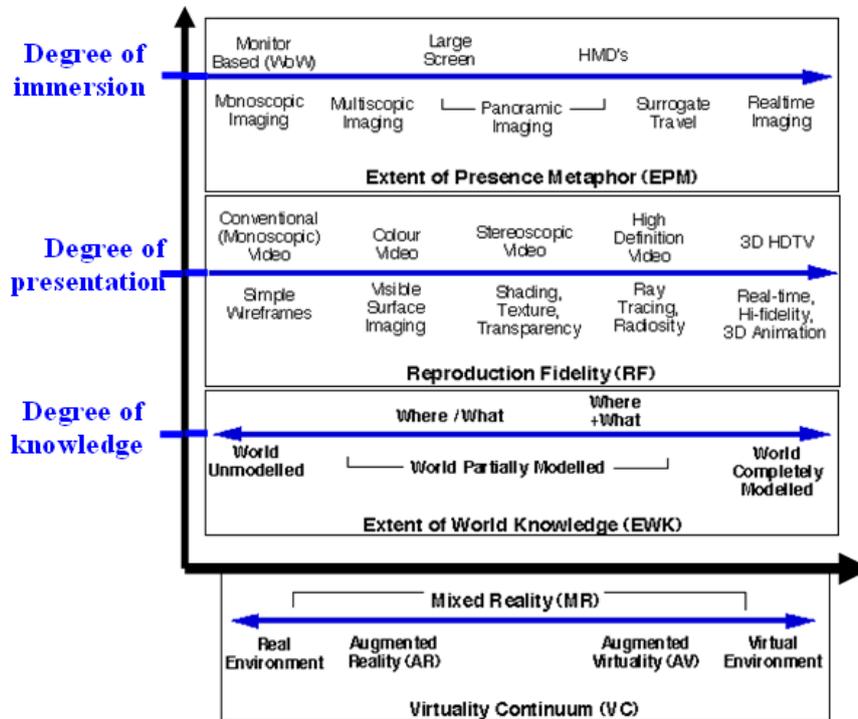


Figure E-5 Extended representation of a "virtuality continuum"

How realistically are we able to display the world? the Reproduction Fidelity. This dimension deals with realism of the projection on the display, in terms of image quality and in terms of immersion, or presence, within the display. This differentiation is based on the device that is used to render the images.

What is the extent of the illusion that the observer is present within that world? They called this the Presence Metaphor. This element refers to the degree of immersion, which range from non-immersive (exocentric environment) to the high-level of immersion (egocentric environments).

In the above description of the continuum, summarized in Figure E-5, the x axis denote the different level of mixed reality and the y axis denote all the elements the application should accomplished. For instance, if we consider a virtual environment, the three values for it are: 1) the maximum for degree of knowledge (full information of the model to be rendered), the highest quality for the presentation (i.e. Real-time hi fidelity, 3D animation, etc.), and finally (3) the highest possible degree of immersion (real time imaging).

Appendix E. Three Dimensional User Interfaces Taxonomy

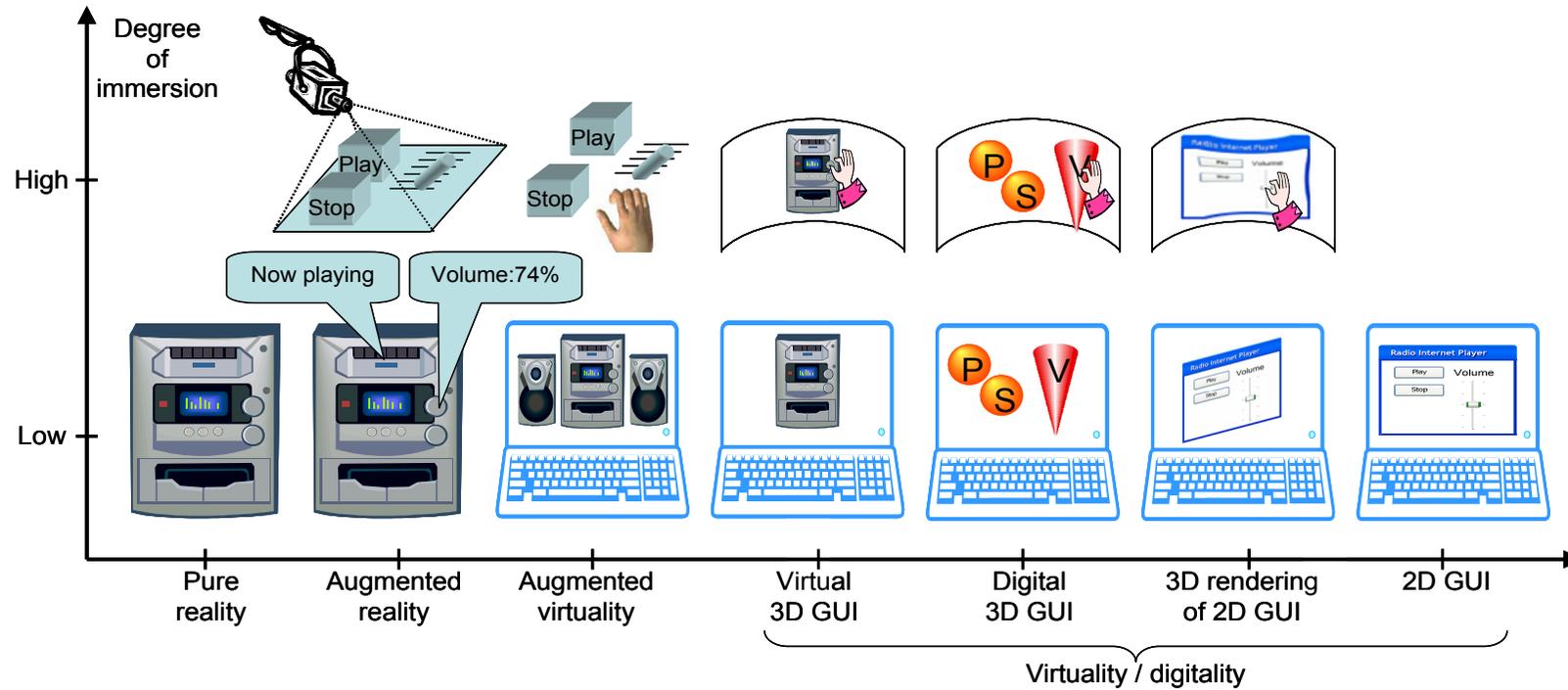


Figure E-6 An extended continuum of user interfaces

2. A Taxonomy for 3D User Interfaces: Extension of the virtual continuum

Milgram and Kishino continuum of mixed reality [Milg94] just point to MR applications but actually the diversification of virtual worlds has more varieties than just MR. Our goal is to explore how we can expand, refine these specification to reach a wider spectrum of 3D user interfaces (UIs) rather than just MR applications. This offers a wide set of options for new developers to identify the type of application that they want to develop.

To introduce and define a wider spectrum of such interfaces while offering different basic or advanced mechanisms and techniques for virtualizing a user interface, we have extended the mixed reality continuum by adding a more continuous range of UIs in the virtual part (Figure E-6) since it can be a 2D UI, a 3D rendering of a 2D UI (whether genuine as in our work or simulated), a genuine 3DUI manipulating 3D objects, and so forth. Moreover, we keep the idea of having dimensions to represent the degree of immersion. We propose just two levels: Desktop immersion (exocentric worlds) when the user is only looking at the screen (desktop virtual UI) or high when the user is really immersed in the system CAVE, HMD in a physical space (egocentric worlds) , similarly to [Poup98] for their interaction techniques taxonomy.

A. Pure Reality

All the kind of displays listed above clearly share the common feature of juxtaposing “real” entities together with “virtual” ones. To differentiate between the two terms they proposed the following description, the “real” object is one that has an actual objective existence, an image that looks real, even could be generated by a computer (non-direct viewing of the object) and is presented in a display or is shown through a HMD, also, a real object has luminosity in its location. On the contrary, a “virtual” object exist in essence or effect but not formally or actually, it cannot be sampled directly and thus it can only be synthesized, this means that an object even that could look real it is not, to clarify this the luminosity of objects do not exist, so the presentation is similar to holograms and mirror luminosity images. More clearly, a “virtual” image of an object is one which appears transparent, that is, does not occlude other objects located behind it.

Pure reality refers to real-world objects of hardware with which we interact, for instance, the stereo system in Figure E-6. In pure reality objects we identify the source of motivation for the development of software interfaces, the innovation of applications, the born of new metaphors. As in 3D we have less restrictions of space as in traditional 2D desktop applications it is possible to create sophisti-

Appendix E. Three Dimensional User Interfaces Taxonomy

cated applications that normally are inspired in a pure-reality objects. Several examples have been developed in this sense; see in Figure E-7 the rotational menu presented in [Wang05].

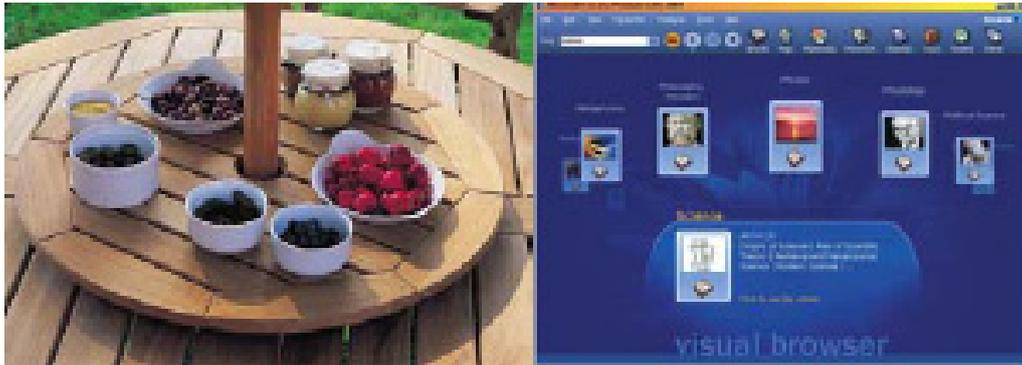


Figure E-7 Three-Dimensional carousels in daily life and computer interfaces

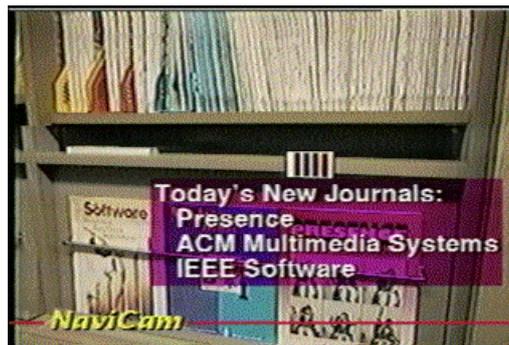


Figure E-8 Rekimoto's NaviCam system and augmented Interaction 1995



Figure E-9 Rekimoto's Augmented Surfaces

B. Augmented Reality

[Milg94] define augmented reality (AR) as “any case in which an otherwise real environment is augmented by means of virtual”. AR interfaces allow users to effectively interact with augmented virtual objects as well as share them with each other in a simple and efficient manner, just as we do it with everyday physical objects. AR is quite appropriate for describing the essence of computer graphic enhancement of video images of real scenes.

In AR applications there are some kinds of Ubiquitous Computing, which change the way in which users interact with computers by providing (virtually) ubiquitous access to services and applications through a large number of cooperating devices. Also, the use of HMD applications are good examples of augmented reality when the see through capability is enabled. Figure E-8 shows an example of an augmented non-immersive application. The real world reminds but through a HMD certain information appears in this case referent to the description of journals that are in a bookcase.

On the immersive-degree of immersion case, the situation changes in the felling of the immersion but the user still interact with physical objects, in this category, we could classify the graspable interfaces, term introduced by Fitzmaurice 1995, [Fitz95]. This kind of applications refers to the projection of user interfaces that use of physical artefacts to control, organize and manipulate digital information. [Reki99] work (Figure E-9), is an example of augmented reality with a high degree of immersion. Images are projected on a table and there is a tracking on the markers so the interaction of the world is tangible, physical interfaces + augmented reality interaction with computing devices.

C. Augmented Virtuality

A virtual word is one that is generated primarily by computer; those represent the augmented virtuality (AV) unlike virtual reality that replaces the physical world, AV enhances the physical reality by integrating virtual objects into the physical world which become in a sense an equal part of our natural environment. The low degree scenario of augmented virtuality is a virtual representation of a user interface that interacts with a physical device. We could use the interface displayed on a screen, on a wall, etc. and using an input device the user is capable to directly operate on the device through the interface as a remote control.

An example of augmented virtuality with high degree of immersion is the work of [Kiy00] (Figure E-10). The user is surrounded by a virtual representation that is projected through the HMD devices, but they also could see and interact with the real-world, as the see through capability is enabled.

D. Virtual 3D GUI

A graphical user interface (GUI) is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use, generally are related to control programs on the computer. This is one of the extensions that we propose to the original continuum. We think that with the development of computer graphics, virtual desktop applications are more frequently used. This is what we called virtual 3D GUI. Notice that all what the user see is generated by the computer and there is no interaction with real world things.

A purely virtual 3D GUI, i.e. with low degree of immersion, consists of a world where virtual objects mimic their real-world counterparts, but displayed on the user's screen. All objects of the UI are virtual and directly operated by direct manipulation of them. The magic mirror of [Gros99] is an example of the use of a mirror metaphor to help the user to see what is occulted in the virtual world.

The high degree of immersion virtual 3D GUI is a graphical representation on a different display, not mandatory, than the traditional screen, a Cave environment could be an option. The interaction with the objects projected in the display is with the user's hands, but they could wear a HMD that recreates their hand. The image plane of [Pier97] (Figure E-12) is an example of this type of applications. The user selects objects projected in a wall with his hand.

E. Digital 3D GUI

The digital 3D GUI is a 3DUI where 3D objects correspond to the tasks (e.g., a sphere to trigger the "Play" function) and the elements (e.g., a cone representing the current volume). The 3D objects used, not necessarily correspond to known or traditional metaphors. The differentiation with the previous definition is that objects are spatial, this means, they are not necessarily attached to a wall, a table or anything in the virtual world. To differentiate the degree of immersion, the only difference is the type of display used to render de virtual world, this means, low degree is the presentation on a screen, and high degree is the presentation on a Cave display.

In this category we also identify innovation, such as that done by [Dach01] with their Collapsible Cylindrical Trees (Figure E-13), an innovative way to present menus in 3D.

Appendix E. Three Dimensional User Interfaces Taxonomy



Figure E-10 Kiyokawa et al. 2000



Figure E-11 Magic Mirror

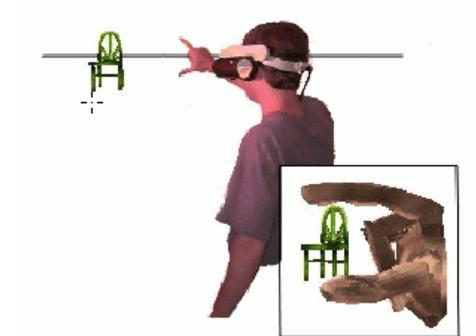


Figure E-12 Image Plane



Figure E-13 Collapsible Cylindrical Trees

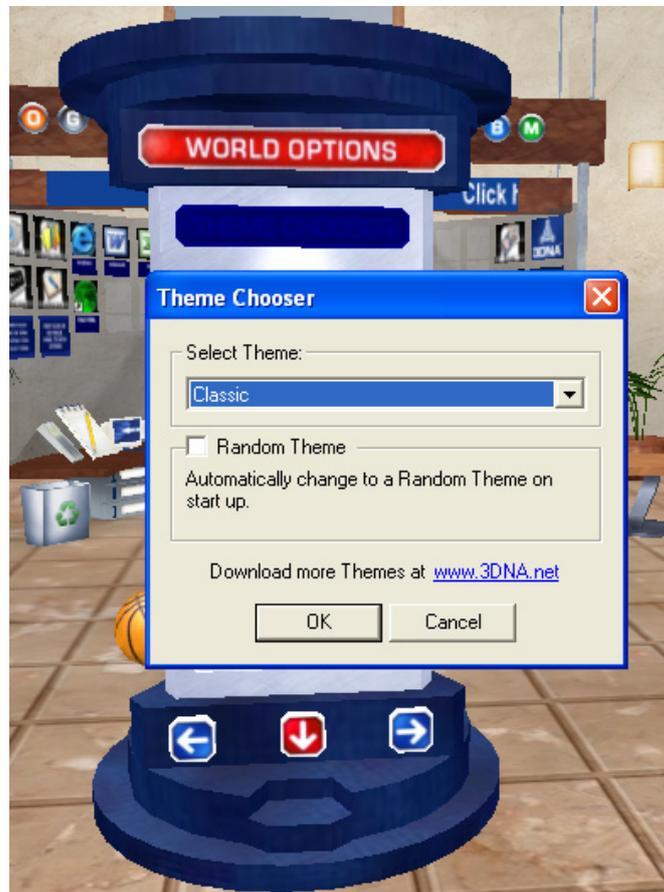


Figure E-14 3DNA desktop application

F. 3D rendering of 2D GUI

A 3D rendering of a 2D GUI in a virtual environment is typically a 3D desktop such as Task Gallery, Windows 3DNA (www.3dna.net), Figure E-14, where traditional windows are manipulated in a virtual environment. To differentiate the degree of immersion, the only difference is the type of display use to render the virtual world, this means, low degree is the presentation on a screen, and high degree is the presentation on a CAVE-based UI where the classical 2D UIs are projected and manipulated by glove and hand recognition techniques.

[Moli05] proposed the VUIToolkit, a set of widget prototypes implemented in both VRML97 and X3D versions that makes possible to map interface elements described at the CUI (Concrete User Interface) level of UsiXML and those that have been included in the toolkit to allow the generation of a FUI (Final User Interface) in a VRML97 or X3D-based 3D environment. This toolkit quickly produces 2D GUIs rendered in a 3D environment, based on UsiXML language

Appendix E. Three Dimensional User Interfaces Taxonomy

(www.usixml.org). The final result of their applications is shown below (Figure E-15), and is in X3D and VRML format. One of the remarkable characteristics of this toolkit is that it transforms the standard plain 2D widgets into a truly 3D representation, not as in 3DNA; see Figure E-14, in which dialog boxes are exactly the same as 2D GUI.



Figure E-15 The virtual laptop with the rendered in VRML97

G.2D GUI

A traditional GUI is projected in 2D, there a lot of examples, such as: Windows, Microsoft Office, Internet Explorer, etc. Also, there a lot of programs, such as visual studio (Figure E-16), Delphi, JBuilder, that are 2D GUI based that serve to develop new 2D GUI Applications.

Appendix E. Three Dimensional User Interfaces Taxonomy

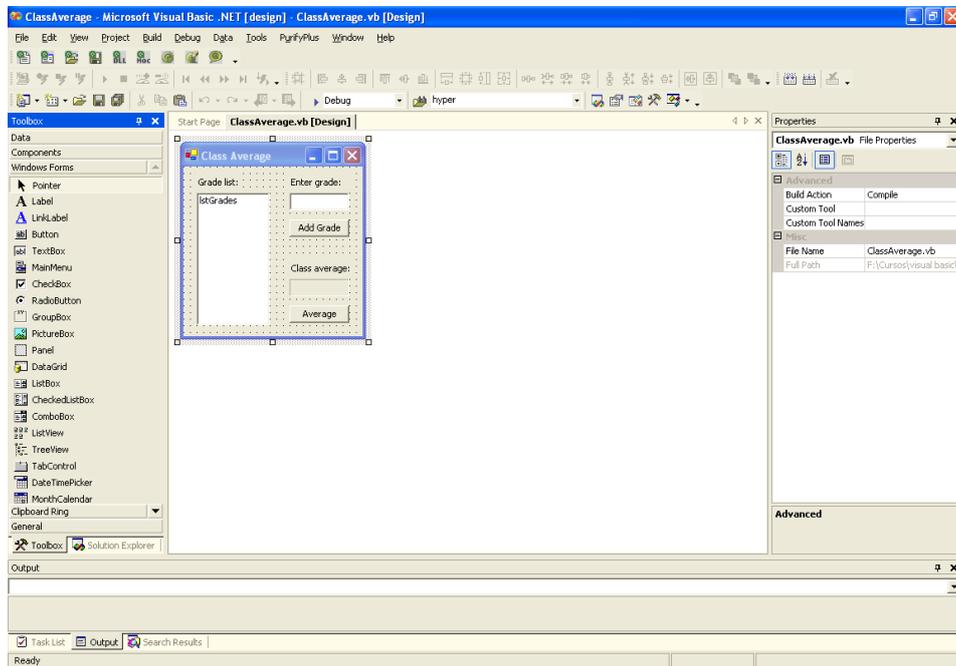


Figure E-16 Visual Studio software tool to develop 2D GUI applications

3. Conclusion

Taxonomies are the results of the impression of their authors and mostly used in their research. Sometimes they are useful for other authors when looking for existing solutions to a well know problem. While there is an agreement that there is no better or bad taxonomy, this depends on what they study [Bowm00], taxonomies could be used to generate a particular classification. Also, later in this dissertation the findings from bowman taxonomy are considered for task patterns. In addition, taxonomy is useful to understand the task in a more detailed level; understand how certain techniques address the task [Bowm00]. The taxonomy proposed here allows the identification of the basic components to create a model-based approach for the 3D user interfaces, as is a way to capture the technical aspects.

In this appendix an extension of Milgram and Kishino taxonomy for mixed reality applications [Milg94] is proposed. The extension incorporates: more levels of virtualization, including the new pushing Web-based virtual applications (using screens to display the virtual world, do not let the impression of immersion).

Used to select widgets the taxonomy showed that not always there is a solution for all the categories of the taxonomy. Then, existing solutions in different levels can be used as source of inspiration. For instance, the taxonomy of a file picker

Appendix E. Three Dimensional User Interfaces Taxonomy

could not have a representation for a mixed reality application but in 2D this object is part of most APIs.

It was identified that the *digital 3D GUI* category of virtualization has not been well exploited. Traditionally, the relevant issues for designers are always on VR content rather than on the system-controls. So as to preserve the already well-known objects used to represent them. The remainder of this work considers *digital 3D GUI* to explore new implementations.

Finally, the taxonomy proposed tries to cover not just the types of VR applications but also two of the most important sources of inspiration. One of the goals of this taxonomy is to provide design ideas. In an scenario in which a developer wants to design a 3DUI, for instance, a ring menu, they want to know the appearance of that widget in 3D. Designers, developers could see how it has been done, if there is a solution, or, if not, how at other levels of the taxonomy authors solved the problem.

Appendix F. State of the Art

1. X3D and VRML

X3D [Web304b] is an open standard for 3D content delivery. It is the next revision of the VRML97 ISO specification. The evolution of the language was not just about using an XML format for VRML but also includes a set of improvements in the semantics of the language and architectural improvements based on past experience using VRML. Even though, it is possible to write X3D code using VRML syntax.

Surprisingly VRML and X3D are not a programming libraries in the strict sense of their definition. As, they do not have an API [Care97], the language was built on top of Open Inventor [Sili03] file format. However, if an author needs more power which means, to develop more interesting application a script is needed. Then we have both: presentation (X3D geometry) and dialog (X3D runtime behavioural descriptions + script).

The set of tools built around X3D covers: editors (RadVRML, openVRML, X3D-Edit), browsers (Xj3D, FreeWRL, SpaceTime3D, Dynamic 3D, Carina), renderers or plug-ins (vivaty, cortona, openVRML, BS Contact of Bitmanagement), and toolkits (vivaty studio). Also, a set tools to import and export from authoring tools such as: 3DMax, Maya or CAD, to VRML97 or X3D format are available.

As X3D and VRML are used for the implementation part of this thesis and more details about the implementation aspects and reused models is discussed in next sections.

2. Toolkit Programming

Toolkit 3D modelling is one of the most used mechanisms used to develop 3D UIs. In this case a toolkit is seemed as the set of basic building elements for graphical User Interfaces that can be implemented whether in a library, such as Open Inventor, or an application framework such as Alice.

Appendix F. State of the Art

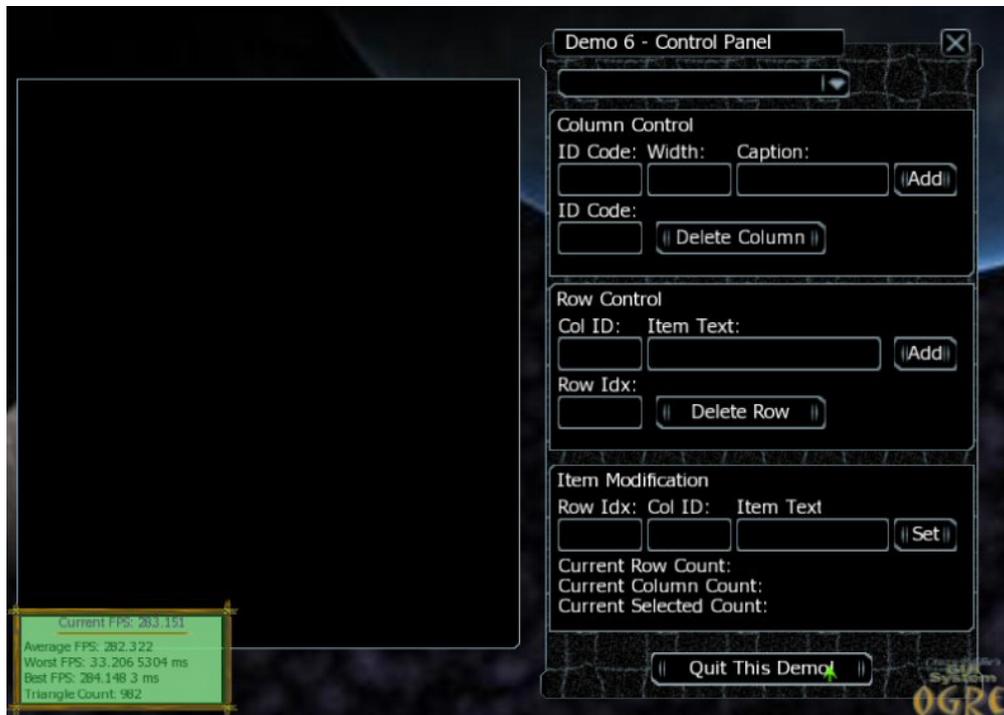


Figure F-1 Game user interface created with Crazy Eddies' toolkit

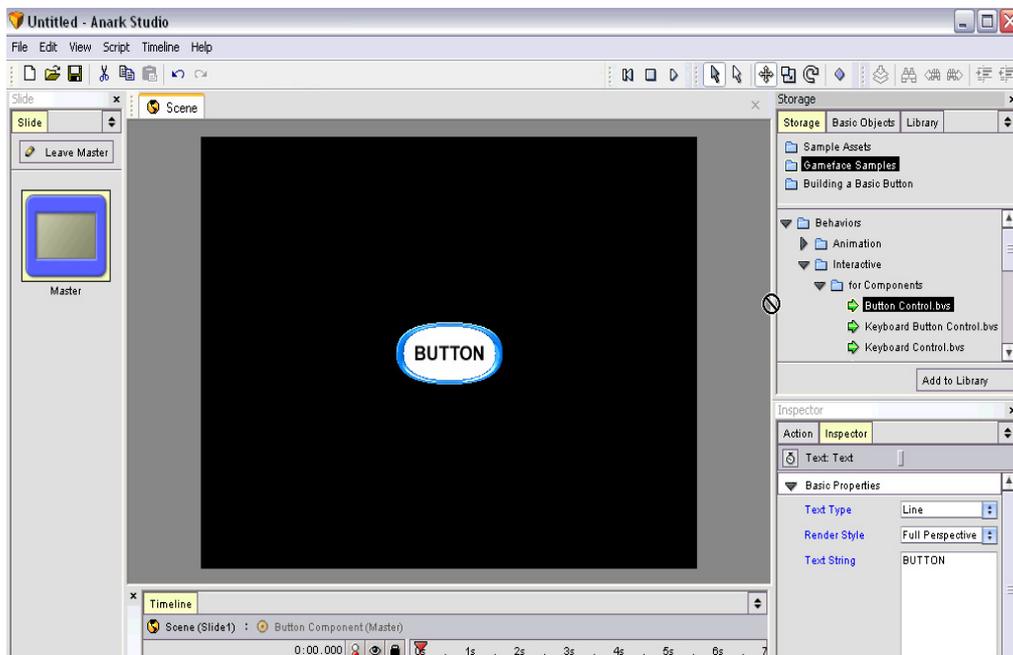


Figure F-2 Anark Studio adding behaviour to a Button

3. Crazy Eddie's GUI System

Crazy Eddie's GUI (CEGUI) System is a free object orientated library, written in C++, and targeted at games developers for a 3D rendering of GUI (3DRGUI) sub-systems [Craz06]. The 3DRGUI toolkit is composed by: button, checkbox, combobox, editBox, Framewindow, listHeader, ListHeaderSegment, multiColumList, MultiLineEditbox, ProgressBar, PushButton, radioButton, ScrollablePane, Scrollbar, ScrolledContainer, Slider, StaticImage, StaticText, ListBox, Thumb, TitleBar, tooltip and window [Craz05]. An example of such 3DRGUI is shown in Figure F-1. Notice that the UI corresponds to a GUI rendered in a video game in 3D.

4. Autodesk Family®

Autodesk® family of software tools for 3D development includes several toolkits for 3d development. This family of software is powerful for designing: Manufacturing solutions, Infrastructure (Civil Engineering & Construction, Electric Utilities, Emergency Response, Mapping & GIS, Public Works, and Transportation) and Building [Auto06a]. The advantage of using this software is that with the plug-ins they provide, it is possible to import/export files from commercial and non-commercial web 3D languages, such as: Java3D, X3D, VRML and O3D.

In Autodesk family of software, among the variety of tools, there are: Autodesk inventor® that helps designers to pass 2D designs to 3D models; this software is very powerful and provides the possibility to create dynamic animations very useful to architects, “*the software is more dedicated to expand AutoCAD® users transitioning to 3D*” [Auto06a]. There are two more oriented tools to design Web 3D which are: Maya ® and 3D Studio Max ®, both are considered the world's most powerfully integrated 3D modelling, animation, effects and rendering solutions [Auto06b]. Maya has been used widely to create movies. Having a lot of open source work around them, including plug-ins to make them compatible with other formats, an informal but accurate list can be found in [Bake06b]. The disadvantage of using this software is its cost which could be a restriction for some developers.

5. Anark Studio

Anark Studio is an application authoring tool for creating interactive 3D content. The product is oriented to design interactive 3D content [Anar06a]. The key features of this software, not limited to, are: it runs 3D presentations on over 90% of existing computers, author drag-and-drop custom widgets such as menus, buttons, and more, Text Object provides crisp, professional and full-featured text

Appendix F. State of the Art

that integrates seamlessly with 3D scenes, even text display text data from external sources like XML.

The 3DUI behaviour is added in using two alternatives. On the first hand, animation time frames can be added, as in any other toolkit. The main advantage in Anark is that more complex behaviours can be added using a mechanism, similar to: on Event if Condition then Action (ECA rules), using a toolbar in Anark editor. The system has a set of actions and events that allows to program interactivity without scripting, as is necessary in VRML or X3D. For instance, in Figure F-2 a predefined behaviour is selected to be added to a 3D button. As there is a script language to support this behaviour mechanism, Lua script (www.lua.org), it is also possible to code directly without using the graphical editor [Anar06b].

Anark uses an intermediary XML file-format which allows the tool to import and export to any other tool using a similar intermediate format, such as COLLADA (<https://collada.org/>). Such intermediate system is useful to integrate application into Web-based systems; also, it provides the possibility to dynamically generate complex Anark Web sites.

6. Alice

Alice [Carn06] is an open source toolkit for defining 3D content based on predefined sets of objects. Is a funny way to learn about object oriented programming so as to handle with 3DUIs. Alice was build on top of Java and Java 3D. Alice applications can be exported into Java 3D and consequently integrate Web-based systems. A browser with the java 3D virtual machine installed would be enough.

Alice editor is divided in four main sections. The scene tree in the top left, in which all the objects attached to the 3DUI. Below, the three tabs to display the properties, methods and functions related to the object selected in the 3DUI tree. On top centre the preview of the world. On the top right the event handler editor and on the bottom right the method and function editor. Both editors work with a drag and drop of properties and a predefined set of events (such as onclick, when world starts) and of control sentences (such as loops, conditionals). A set of predefined actions are assigned to each 3D object.

7. Google SketchUp

Google SketchUp (<http://sketchup.google.com>) by Google is a 3D modelling software in which the modelling is based on the analogy of 2D drawings used by architects to depict 3D objects in 2D. These 2D objects (rectangles, circles or any other polygon drawn with the pencil) are attached to any pair of the three axes, for instance, a rectangle is created in the x and y axis and then you can add volume by drawing the rectangle in the z axis, using the volume tool. Even that the

Appendix F. State of the Art

idea of using 2D shapes to generate 3D objects is not new, Autodesk software families used this feature to import CAD models to build 3D objects (see section 4), the new approach created by Google is assumed to be natural and simple to use even by non experts, operations such as: extrude, binary operations, are supported implicitly.

One of the innovative features of the tool is its capacity to use an image as a source to draw 3D models. Once the 3D model is finished the original image can be used to add textures and then an enough detailed object can be obtained using this tool. For instance, in Figure F-3 the cathedral of Puebla in Mexico has been build based on a photo. More images from different corners, perspectives, are desirable in order to get a precise texture and size of the object you are modelling. Also clean images are preferable, i.e., no overlapping of objects that are not part of the image.

The original purpose for SketchUp was to created content for Google Earth (<http://earth.google.com/>). However, a new API, called O3D (<http://code.google.com/apis/o3d/>), has been released that can be used to add behaviour to 3DUIs. SketchUp is in the category of content development and in combination with O3D is a good option to design 3DUIs.

8. Vivaty Studio

Vivaty Studio™, Figure F-4, by Vivaty Inc. (www.vivaty.com) is an authoring tool for creating real-time Web 3D. It uses VRML and X3D as its core format but it supports the import and export to others standard formats, such as: Google Sketchup, Blender, Autodesk 3ds and Maya. Vivaty handles not just presentation aspects but also animations and basic standard from X3D specification. In addition, it uses the ECMAScript language to specify more complex behaviour. Objects can be modelled by using the basic shapes (sphere, cube, cone and cylinder) or using one of the extrude objects to draw series of points connected by lines to form 2D shapes that then are projected to build a 3D shape. The main characteristic to define objects in Vivaty Studio is the mesh editor. Every object can be transformed into a mesh, i.e., a powerful tool to edit shapes in detail. One last interesting feature is the support of single Mesh Avatars. There are a set of predefined avatars that can be added to the scene in Vivaty Studio.

Appendix F. State of the Art

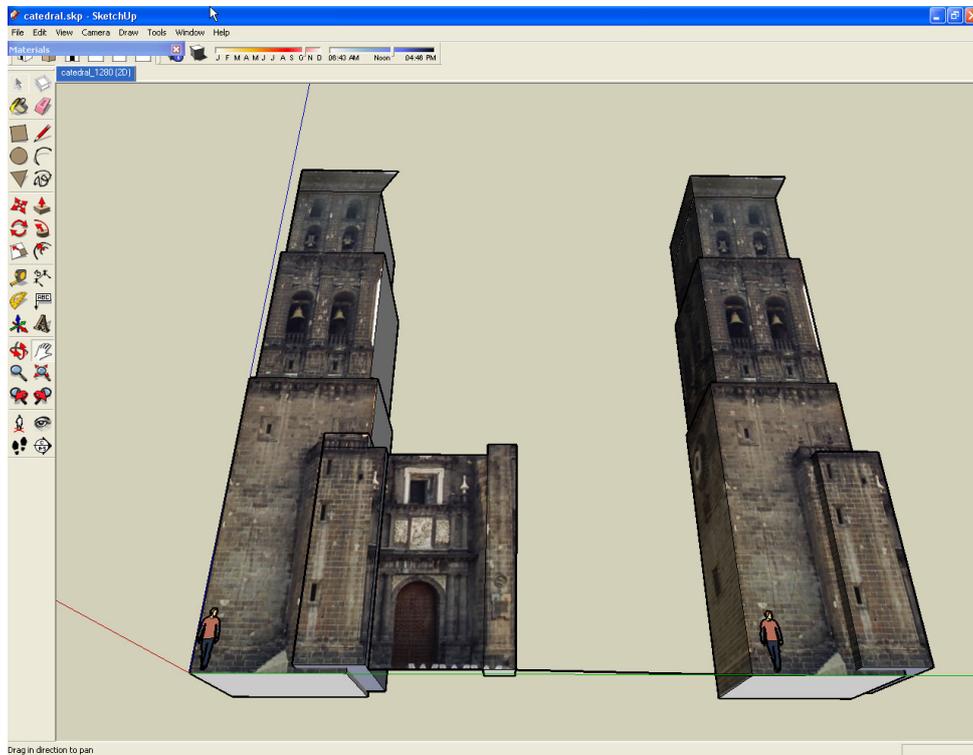


Figure F-3 Puebla's cathedral designed in Google SketchUp

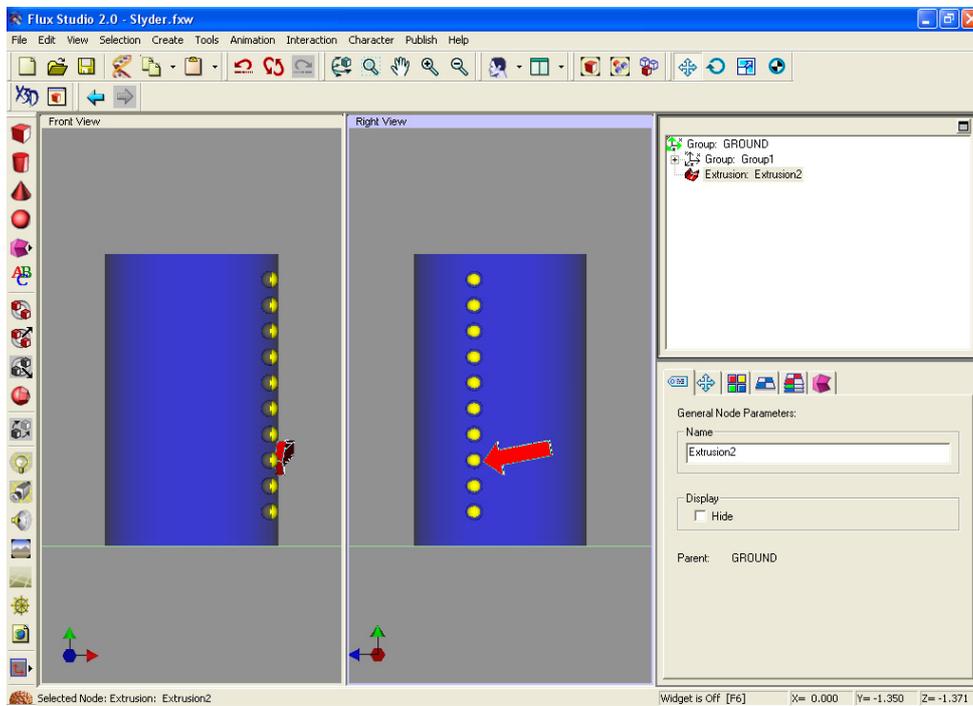


Figure F-4 Vivaty Studio

Appendix F. State of the Art

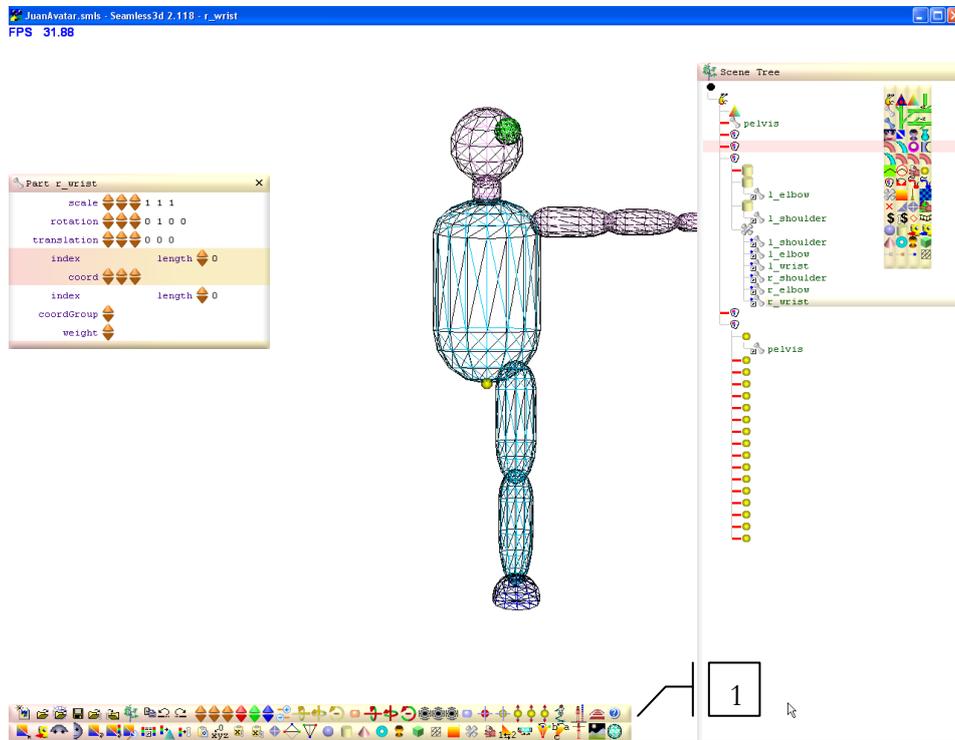


Figure F-5 Seamless3D an avatar studio



Figure F-6 OGRE game rendering

9. Seamless3D

Seamless3d (<http://www.seamless3d.com>) is tool to make 3D animated content for the web. Even that any object can be designed in the tool, its main purpose is to design avatars. This tool is compatible with VRML/X3D formats. The toolbar is composed of 90 buttons (1 in Figure F-5) so the UI of this system is not friendly. Avatars are important for 3D interaction as they add some realism. However, in this researcher interaction with other users is out of the scope.

10. Game engines

The game manufactures are the leading industry for 3DUI development. Most of existing work is dedicated to create games. The game market includes the web as well, as it was revealed by some articles from the BBC. The market for massively multiplayer online games (MMOGs), as is called, is more than \$1bn, BBC online journal (<http://news.bbc.co.uk/2/hi/technology/6470433.stm>). Some of the game engines, not limited to, are: Crystal Space 3D (www.crystalspace3d.org), Open Scene Graph (www.openscenegraph.com), Irrlicht 3D (irrlicht.sourceforge.net), Panda3D (panda3d.org) and XNA.

OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, 3D engine written in C++. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes [Ogre05]. OGRE just provides a world-class graphics solution; for other features like sound, networking, AI, collision, physics etc, other libraries are required. Many experienced game developers have expressed their approval of this approach, because there are no inbuilt constraints, [Ogre05]. In Figure F-6, an example of the rendering made with OGRE, notice that the components regarding the User Interface remains in 2D.

11. Three Dimensional Desktop Systems

Several 3D desktop replacements for Microsoft Windows XP exist. These environments are very powerful for their manipulation of windows in 3D, but they are not intended to render 2D UIs with 3D effects.

SphereXP (<http://www.hamar.sk/sphere/>) offers a new way to organize objects on the desktop. The Sphere becomes the new desktop space. The user is exactly in the middle of it and all objects surround him. Objects can be moved around the sphere according to some rules. It is possible to bring objects closer to the view point and send them back.

Appendix F. State of the Art



Figure F-7 SphereXP three dimensional desktop



Figure F-8 Desktop in 3DNA

Appendix F. State of the Art

The 3DNA (www.3dna.net), Figure F-8, is a different and unique desktop replacement. The desktop is a physical representation of a house where the user navigates from one room to another to go to different application groups (media, browser, games). The movement and navigation within a 3DNA Desktop is identical to a 3D video game and is also fully customizable by the user (short cuts can be easily specified to a specific place in the world). In addition, 3DNA provides ways to organize files, folders, and applications-direct access, in different objects such as the wall or drawers. Furthermore, 3DNA provides an interesting capability, which was not used before Google Chrome, to speed-surf dozens of websites at a time. The favourites section has a screenshot of the web site, which we can zoom in/out, this offers the user more means to remember a booked page.

Similarly, SUN initiated the Looking Glass Project (www.sun.com/software/looking_glass/), a 3D desktop environment for Linux workstations. It is an open source development project that supports running unmodified existing applications in a 3D space, as well as APIs for 3D window manager and application development. The library for 3D application development is available for Linux, Solaris and Windows. In Figure F- 9 two different desktops, each with a window opened, with different number of items in the task bar are shown.

It is possible to create your own version of 3D desktop. For instance, 3DNA offers this option of having access to their API when buying a license. The open-source project looking glass by definition can be adapted to any particular need. However, any ad hoc version does not guarantee its compatibility with future version of the project.

12. Rendering 3D Web browsers

Clara (<http://www.spatialknowledge.com/projects/clara/>) is a 3D web browser that lets walk, fly, or jump through a virtual world where all the objects are usable, interactive web-page. In Clara it is possible to read the pages as traditionally interacting in 2D, i.e. scrolling, clicking, or hearing is possible. The screenshot of Figure F-10 shows the presentation of Clara rendering multiple pages at once. Clara is implemented as a Windows program in versions for 3D boards running OpenGL or DirectX.

Appendix F. State of the Art

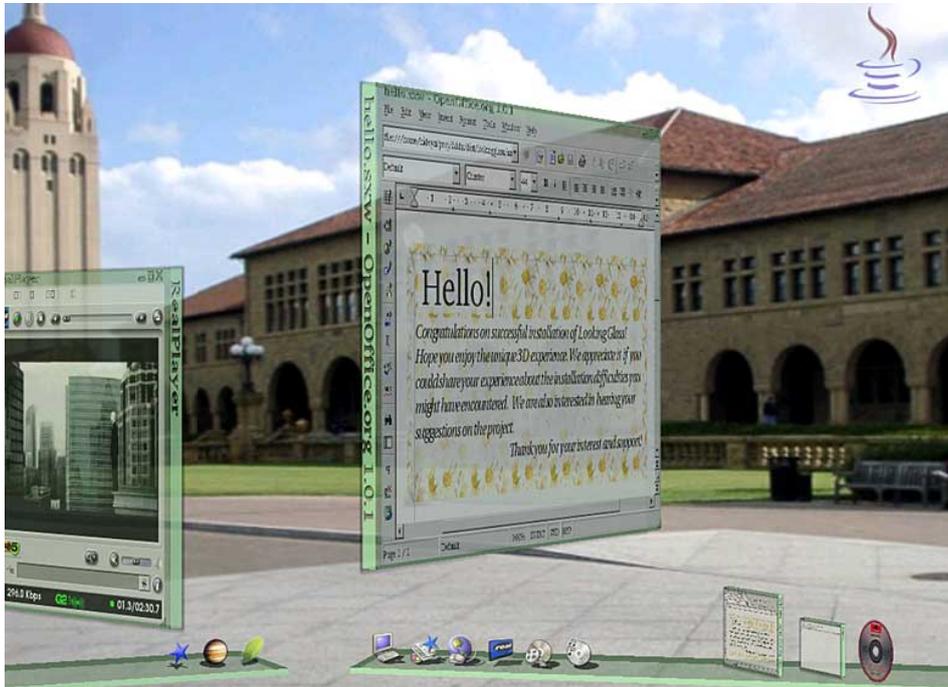


Figure F- 9 Looking glass Desktop



Figure F-10 Clara Web Browser

Appendix F. State of the Art

A similar approach to Clara web browser is Space Time (<http://search.spacetime.com/>). In Figure F-11 the *images* view of this browser is shown. The swine flu search shows a set of images that are displayed one by one in the centre but the user still have access to previous and next images.

This idea of presenting windows is explored in Chapter 3. More than switching from 2D to 3D, these web browsers just use frames to show the preview and when clicking on a window then the web browser is used to get into the web site.

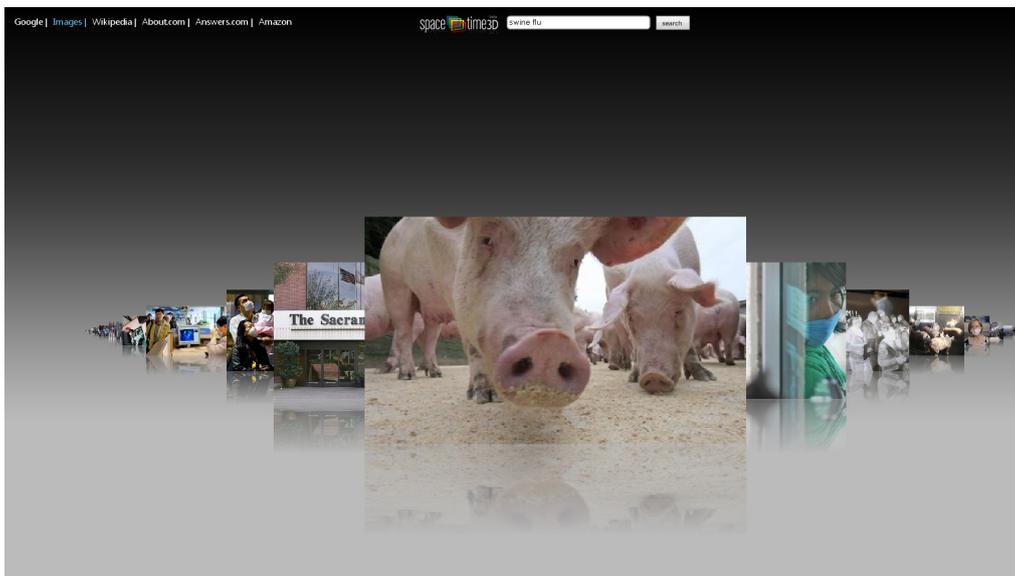


Figure F-11 Space Time Browser



Figure F-12 Exit reality

13. Web Virtual Worlds

Recently, web social networks uses, among other tools, 3D web based applications. The most famous is Second Life by Linden Labs (<http://secondlife.com/>). However it is not the only one, there some others such as: IMVU (www.imvu.com), VR4ALL (www.vr4all.net) or Vivaty that is a plugging for two social networks AIM (<http://gallery.aim.com/detail/551>) and Facebook (<http://apps.facebook.com/vivatybeta/>).

The interest on describing these 3D visualization and interaction in social networks is to show the growing interest of web 3D. As well, it is interesting to analyze the different mechanism used for interaction, apart from the chat, in these virtual worlds. While the main purpose in to make interaction between avatars more fun, there is not enough effort in the interaction objects used. This can be seen in Figure F-12 where an avatar navigates through the virtual world Exit Reality (www.exitreality.com). On the right a very nice television renders a video from YouTube. Curiously, not real 3D buttons are available to control de video (stop, play, pause, volume).

It is interesting to analyze the different metaphors used in these virtual worlds. One major drawback is that developing something for these worlds is either not public available or time consuming, basically these worlds has everything a user might need and nothing else could be added.

Appendix G. Task patterns

1. Navigation pattern

In [Bowm01a] navigation is defined as the composition of two concurrent tasks: travel and Wayfinding. Similarly [Tan01] define three tasks which are: knowledge (Wayfinding), search (travel) and inspect (system control). We consider the proposal of [Bowm01a], as system control in our opinion is not part of the navigation task itself. The user can navigate thought controls that can be defined in the travel task.

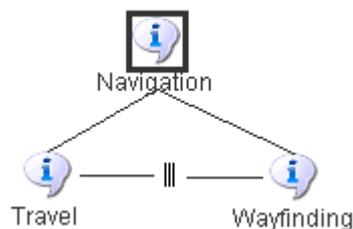


Figure G-1 Navigation pattern

2. Travel pattern

Travel is the motor component of navigation and just refers to the physical movement from place to place. The described Interaction techniques in [Bowm01a] for travelling are:

- Gaze-directed steering using head tracking [Mine95]
 - Indicate direction
 - User move head toward the desired direction
 - System track direction
 - Translate to view point selected
- Pointing using hand tracking [Mine95], [Bowm97b]
 - Indicate direction
 - User move arm toward the desired direction
 - System track direction
 - Translate to view point selected
- Map-based [Bowm98]
 - Indicate direction

Appendix G. Task patterns

- Select icon
- Drag icon
- Release icon
- Translate to view point selected
- Grabbing the air [Mape95].
 - Indicate direction
 - Select position (pinch or click on it)
 - Translate to view point selected
 - Stop selection
 - Release button or stop pitching

The pattern for travelling is depicted in Figure G-2. As explained above, several interaction techniques allows the travel task but all of them considers indicate position and to translate the view.

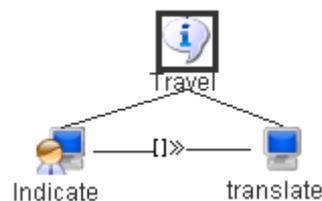


Figure G-2 Travel Pattern

3. Wayfinding

Wayfinding is one of the two pillars when navigating [Krui00], represent the cognitive or the decision-making component for user to define their path through the virtual world. [Krui00] identify different Wayfinding tasks:

- Extract Information
 - User position
- Build up spatial knowledge
 - World structure
- The user uses the spatial knowledge to make a decision



Figure G-3 Wayfinding Pattern

4. Select Pattern

The *Select* task is simply the specification of an object or a set of objects for some purpose. In [Bowm01a] four interaction techniques are defined to do this task in 3D, they are:

- Virtual Hand. The most common technique is the virtual hand metaphor that is the representation of the hand to touch objects as we do in the real world. There are two varieties exist, with augmented virtuality (without haptic feedback) or augmented reality (with haptic feedback).
 - *Indicate* position
 - Move hand
 - *Identify* Intersect/Collision
 - Hand position with objects
 - If any intersection then *Select*
- Ray-Casting is another common technique that uses the metaphor of a laser pointer, an infinite ray extending from the virtual hand [Mine95].
 - *Indicate* position
 - User Move hand
 - System Determine Ray Direction
 - *Identify* Intersect/Collision
 - If any intersection then *Select*
- Sticky finger [Pier97] is a technique that considers the occluded objects, with the virtual hand, that users want to reach.
 - *Indicate* position
 - User Move head
 - User Move hand
 - System Determine Ray position by subtracting hand position
 - *Identify* Intersect/Collision

Appendix G. Task patterns

- If any intersection then *Select*
- Go-go. The go-go interaction technique [Poup96], inspired in inspector Gadget, the cartoon, who had the ability to extend its arm to reach objects. This technique introduces a non-linear mapping between arm extension and virtual hand position.
 - *Indicate* position
 - User Move hand
 - System Determine hand position based on its extension
 - *Identify* Intersect/Collision
 - If any intersection then *Select*

The select pattern is shown in Figure G-4. Notice that the pattern consider the abstract task of indicate, several means can be used to do so, as pointed in this section by means of virtual hands for immersive VR applications, but is general enough to let open the possibility that a 2D based menu can be use to indicate the object. Information passes from this task to the system that has to identify if there is an object that collides with the indication made by the user, if so it will, mark the object.

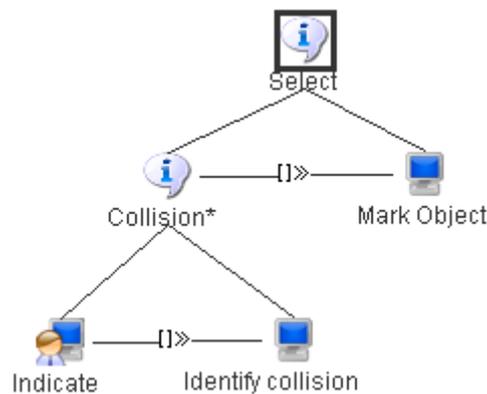


Figure G-4 Select Pattern

5. Manipulation pattern

There is a string link between the manipulation task and the selection. An object cannot be manipulated if in some way has not been previously indicated. The manipulation task is, in some cases, linked with the selection technique. Special care must be considered when an object is released, what is going to be its position, when using a technique, such as go-go. In Figure G-5 we show the manipulation pattern. In [Bowm01a] four techniques are described, which are:

Appendix G. Task patterns

- The virtual hand, used in all the family of hand techniques (go-go, ray-casting, any arm extension).
 - *Select* Object
 - Attached object to the hand
 - *Transform* object
 - *Release* object, depending on the technique the position of the object will be calculated.
- The Hand-centred Object Manipulation Extending Ray-Casting (Homer) technique [Bowm97a].
 - *Select* using ray-casting
 - *Translate* the hand to the selected object
 - *Transform* object
 - *Release* object.
 - *Translate* the hand back to its normal position
- Scaled-world grab is a technique that is related to the occlusion techniques for selecting.
 - *Select* using occlusion technique
 - *Translate* the user or world to the selected object
 - *Transform* object
 - *Release* object.
 - *Translate* the user or world to their previous position
- World-in-miniature (WIM) technique [Stoa95; Paus95] uses a small “doll house” version of the world that allows the user to do indirect manipulation.
 - *Select* Object using any virtual hand technique
 - Attached mini-object to the hand
 - *Transform* object
 - *Release* object, depending on the technique the position of the object will be calculated.

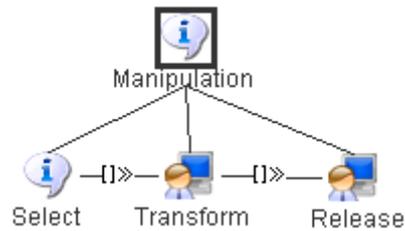


Figure G-5 Manipulation Pattern

6. System control pattern

This pattern is quite complex as there are a wide range of operation that deals with system control. Deals with menus, buttons, speech, tracking, input devices known and new ones. Normally these techniques involve a sort of selection technique [Bown01a], but contrary to the select objects when a system control is selected some kind of feedback must be provided to the user. In Figure G-6 we show a simplified system control pattern. Further investigation will be conducted.

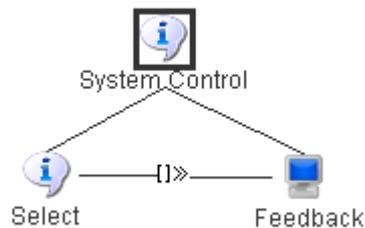


Figure G-6. System Control Pattern.