

A Generalised Framework
for the Design and Construction
of Integrated Design Systems

Robert Wilton Amor

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

Doctor of Philosophy in Computer Science

University of Auckland

May 1997

Abstract

The building industry employs a significant percentage of the workforce of any country, and encompasses a considerable proportion of a country's GDP. Despite that, IT tools used in the design and management of a building project are still fairly crude. Many projects have been undertaken to develop IT-based solutions to support the architecture, engineering, and construction domains (A/E/C), but little effort has gone into the tools required to support these development activities. This is the area in which this thesis concentrates.

To develop a schema representing some subsystem of a building it is necessary to have support tools which enhance the modeller's environment. The current state of the art, a replicated paper based approach, is ineffective at guaranteeing the consistency and validity of large schemas. In this thesis, a more appropriate environment is developed and demonstrated. This provides multiple overlapping views of the developing schema, with guaranteed consistency between all views, the ability for many modellers to work on the schema, and links to related aspects.

The array of schemas being developed for the A/E/C domains contain overlaps of information, though often in different representations. To enable the full use and correct transfer of information between schemas, mappings between their representations need to be defined. This thesis develops a comprehensive mapping language which describes bidirectional mappings between schemas. An automated system has been constructed which can take a mapping specification and manage the updates and consistency of data in models corresponding to the mapped schemas.

To manage the development of environments described above, as well as the finished integrated environments proposed, it is necessary to manage and control the supported processes. A notation is developed to allow this control to be defined, and an implementation is provided to demonstrate how a project can be managed.

The end result of the thesis is a set of notations and associated tools which support all aspects of the development and implementation of integrated design environments. The resultant development environment greatly raises the level of support for developers over that offered by current tools, for all aspects of specification, consistency, testing, validation, implementation, and coordination between developers.

Acknowledgments

I would like to thank my supervisor, Dr John Hosking, for his efforts in keeping me on track, his enthusiasm for the ideals that are encompassed in this research, his support in funding rounds, and for helping with yet another paper.

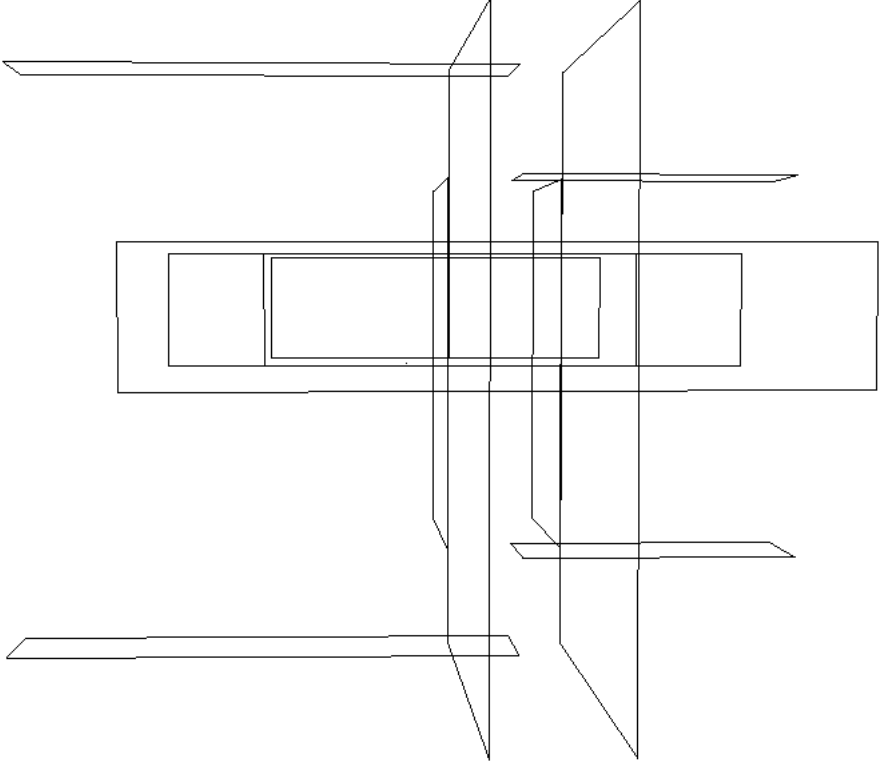
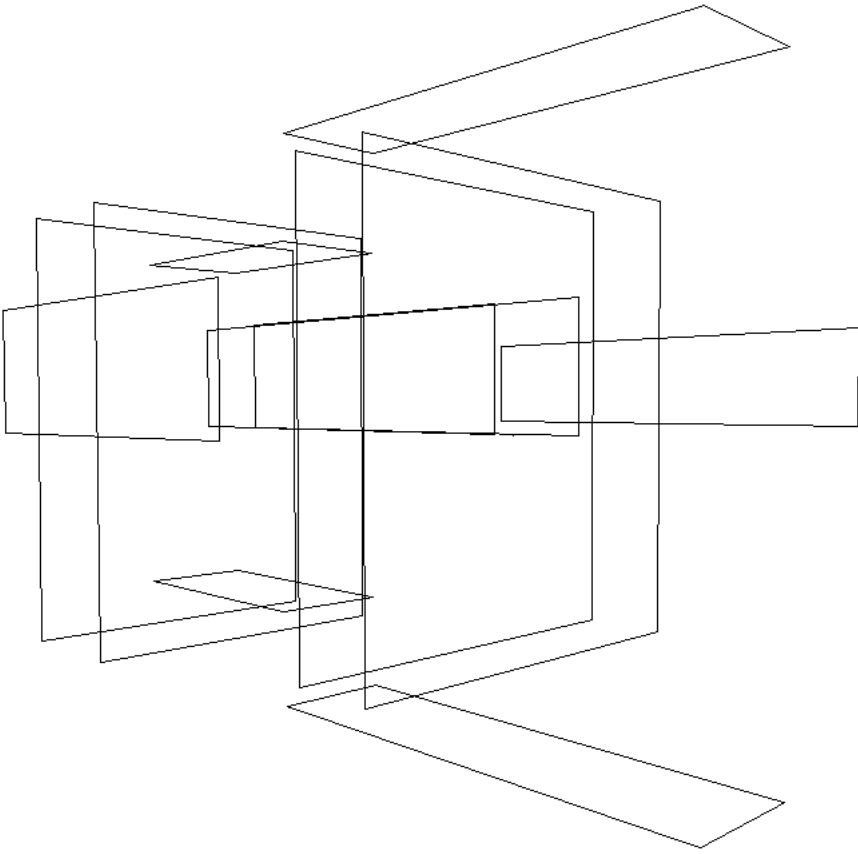
Thanks to my wife Kath for her love and support and making sure it all happened. Now it is her turn and I can get my own back! Some thanks go to our two lovely kids, the arrival of Sylvia (2/2/96) almost got me finished but then made it harder, then the arrival of Roman (21/4/97) almost got me finished again, thankfully Kath's parents have been here for the last month and enabled me to finish this thesis without abandoning Kath to all the child-minding. A word of warning, never leave university with two weeks work left on a thesis. A year and a half later, two children later, and another country later I am just thankful to get this thesis out the door. Also thanks to parents and siblings for their support and encouragement over the years, yes, I know, it is finally time to get a job.

Financial support was provided by a University of Auckland PhD scholarship, a Department of Computer Science scholarship, and a working spouse, many thanks to all of them. I would also like to thank the department for their unfailing support with travel funds for conference trips. The COMBINE group partially sponsored my stay at TU Delft for 6 months and travel costs were partially covered by a grant from the University of Auckland Graduate Research Fund. I found this stay of immense value to my work and thank the groups involved for making it possible.

Thanks also to the PhD students and staff in the department for lively conversations and ears willing to have ideas bounced off. Special thanks to John Grundy, Paul Qualtrough and Achim Schneider, I will get you guys juggling yet! I found the environment at TU Delft very stimulating, you guys have a great group there. Special thanks to Marcel Verhoef, Wouter Rombouts, Godfried Augenbroe, Roel Schipper, and all the others in the group, for making us feel so welcome during our stay and introducing us to the Dutch way. We are still hooked on stroopwafels and wickedly strong coffee!

Thanks to the EU funded COMBINE group for taking the time to listen to this outsider's views. I found the project very stimulating and the flow of ideas exhilarating. This thesis started with a two month conference and university tour of the USA. Thanks to the following groups for making me feel so welcome and sparing the time to explain your projects: LBL (Greg Ward, Steve Selkowitz, Bill Carroll, Fred Winkelmann, and Kostas Papamichael), UCB (Alice Agogino), Stanford (Mark Clayton and Paul Teicholz), Cal Poly (Jens Pohl), UCLA (Chuck Eastman, Hisham Assal, and Scott Chase), MIT (Duvvuru Sriram and Albert Wong), CMU (James Garrett and Skip Van Wyk), University of Michigan (James Turner), University of Wisconsin.

Incorrect Mappings as Artwork



Contents

Abstract	i
Acknowledgments	iii
List of Figures	xiii
List of Tables	xvii
1 Introduction	1
1.1 The Problem Domain	1
1.2 Related Research	3
1.3 COMBINE Project	4
1.4 Underlying Framework	6
1.5 Research Objectives	9
1.5.1 Formalism to specify mappings	10
1.5.2 Formalism to specify project and flow of control	10
1.5.3 Comprehensive tool set to support model specification	10
1.5.4 Inter-model mapping utilising mapping specifications	10
1.5.5 Control system utilising project and flow of control specification	11
1.6 Description of Example	11
1.6.1 The Snart language	12
1.6.2 PlanEntry	13
1.6.3 FaceEditor	15
1.6.4 VISION-3D	16
1.6.5 ThermalDesigner	16
1.6.6 IDM	17
1.6.7 Correspondences between models	17
1.7 Outline of Thesis	19
2 The Project Development Environment	21
2.1 Structure and Requirements	22
2.2 Schema Modelling and Development	23
2.2.1 Requirements of a schema modelling environment	24
2.2.2 IDM schema	25
2.2.3 DT schemas	25
2.2.4 Actor schemas	26
2.2.5 Related schema modelling environment research	26
2.2.6 Approach to a schema modelling environment	27
2.3 Inter-schema Relationship Modelling	28
2.3.1 Requirements of an inter-schema relationship definition language and modelling environment	29
2.3.2 Related inter-schema relationship modelling languages	30
2.3.3 Related inter-schema relationship modelling environments	31

2.3.4 Approach to an inter-schema relationship modelling language	32
2.3.5 Approach to an inter-schema relationship modelling environment	32
2.4 Design Tool Environment Modelling	33
2.4.1 Requirements for design tool environment modelling	33
2.4.2 Related design tool environment modelling work	35
2.4.3 Approach to a design tool environment modelling system	35
2.5 Project Definition	36
2.5.1 Requirements for a project definition notation and development environment	36
2.5.2 Related project definition notation work	37
2.5.3 Related project definition environment work	37
2.5.4 Approach to a project definition notation	38
2.5.5 Approach to a project definition environment	38
2.6 Project Development Environment Summary	39
3 Schema Modelling and Development	40
3.1 Introduction	40
3.1.1 Requirements for schema development	41
3.1.2 Schema specification languages	42
3.1.3 The EXPRESS and EXPRESS-G languages	43
3.2 Schema Development in the EPE Environment	44
3.2.1 Functionality offered by the EPE environment	45
3.2.2 Using the EPE environment	52
3.2.3 Implementation of EPE in the MViews framework	53
3.2.4 Internal schema representation in EPE	57
3.2.5 Summary of EPE functionality	58
3.3 Generic Schema Database Definition	58
3.4 Appraisal of Schema Modelling	59
4 Inter-Schema Relationship Modelling	63
4.1 Mapping Types	64
4.1.1 Structural mapping types	64
4.1.2 Semantic mapping types	66
4.1.3 Mapping language requirements	67
4.2 Mapping Definition Languages	69
4.2.1 EXPRESS-M	70
4.2.2 EXPRESS-V	72
4.2.3 EXPRESS-C	74
4.2.4 Transformr	75
4.2.5 EDM-2	76
4.2.6 KIF	78
4.2.7 Superviews	78
4.2.8 RDBMS views	79
4.3 Summary of Inter-Schema Relationship Modelling	80
5 The View Mapping Language (VML)	82
5.1 Mapping between Schemas	84
5.2 Mapping between Classes	86
5.2.1 Entity names and keys	86
5.2.2 Inheritance of <i>inter_class</i> definitions	88
5.2.3 Invariant specification	89
5.2.4 Initialiser specification	90
5.2.5 Equivalence specification	91
5.2.6 Mapping equations	91
5.3 A Graphical Notation for VML	98
5.3.1 Graphical icons of VML-G	99
5.4 Appraisal of VML	101

6 Mapping Modelling and Development	106
6.1 Introduction	106
6.1.1 Requirements for mapping development	107
6.2 Mapping Development in VPE	108
6.2.1 Functionality offered by the VPE environment	108
6.2.2 Using the VPE environment	116
6.2.3 Implementation of VPE in the MViews framework	118
6.2.4 Future connections between VPE and EPE	120
6.3 Management of Mapping Definitions	120
6.3.1 Name resolution	121
6.3.2 Schema modification	122
6.4 Generic Mapping Database Definition	122
6.5 Appraisal of Mapping Modelling and Development	123
7 Project Modelling	126
7.1 Introduction	126
7.1.1 Requirements	129
7.1.2 Structure	131
7.2 User and Function Modelling	132
7.3 Flow of Control Modelling	133
7.3.1 Set theoretic background for flow of control	133
7.3.2 Flow definition	137
7.4 Appraisal of Project Specification	141
8 The Project Testing and Implementation Environment	145
8.1 Structure and Requirements	146
8.2 Schema Instance Development	147
8.2.1 Requirements of a schema instance maintenance system	147
8.2.2 Related schema instance maintenance systems	148
8.2.3 Approach to a schema instance maintenance system	149
8.3 Mapping Handler and Controller	149
8.3.1 Requirements of a mapping handler and controller	149
8.3.2 Related mapping handler and controller work	150
8.3.3 Approach to a mapping handler and controller	151
8.4 Design Tool Connection	152
8.4.1 Requirements for a design tool connection system	152
8.4.2 Related design tool connection systems	152
8.5 Flow Handling	152
8.5.1 Requirements of a flow of control manager	153
8.5.2 Related flow of control manager work	153
8.5.3 Approach to a flow of control manager	154
8.6 Project Testing and Implementation Environment Summary	155
9 Schema Instance Management	156
9.1 Requirements for Instance Management	156
9.2 Instance Management Systems	157
9.2.1 EPE: an instance construction and browsing system	157
9.2.2 InSTEP: a graphical instance browser	159
9.2.3 SmartQuery and the ObjectViewer	160
9.2.4 Reflex: an object-oriented CAD system	162
9.3 Appraisal of Schema Instance Management	163
10 Mapping Controller	164
10.1 Data-Store Modification Records	165
10.2 The Mapping Controller	169
10.2.1 Transaction-based mapping manager	171
10.2.2 Automatic mapping manager	172
10.3 Performing a Mapping	172
10.3.1 In preparation to map	173
10.3.2 The first mapping between two stores	173

10.3.3	Consideration of modification types	174
10.3.4	Determining combinations of objects from an <i>inter_class</i> header	175
10.3.5	Four pass mapping process	179
10.3.6	Mapping a new combination to the other data-store	180
10.3.7	Procedures followed when a new object is created	181
10.3.8	Tracking objects created and referenced in mappings	182
10.3.9	Mapping the deletion of an object	182
10.3.10	Mapping the modification of an object	183
10.3.11	Evaluating, or re-evaluating affected equations	184
10.4	Appraisal of Mapping Controller	186
11	Flow Handling	189
11.1	Requirements for Flow Handling	189
11.1.1	Project manager requirements	190
11.1.2	Actor requirements	190
11.2	The Exchange Executive	191
11.2.1	Simulation of flow of control	191
11.2.2	Representation of design tool invocation	197
11.2.3	Project manager interface	198
11.2.4	Actor interface	201
11.3	Appraisal of Flow Handling	203
12	Conclusions	205
12.1	The Project Development Environment	206
12.2	The Mapping System	208
12.2.1	Additional applications of the mapping language	209
12.3	The Flow of Control System	210
12.4	Future Work	212
12.4.1	Tighter system integration	212
12.4.2	Distributed environment	213
12.4.3	Wider incorporation of project aspects	213
12.4.4	Formal definitions of the mapping domain	214
12.4.5	Alternate mapping language implementations	214
12.4.6	Incorporation of measure tools	215
12.4.7	Dissemination and exploitation	215
Appendix A.	The View Mapping Language	217
A.1	VML Syntax	217
A.2	VML Graphical Notation	219
A.3	VML Comparison to other Notations	220
A.3.1	Comparison to database operators	220
A.3.2	Comparison to Motro virtual integration operators	222
Appendix B.	Project Specification Language	227
B.1	Project Model Transfer Syntax	227
B.2	Project Modelling Graphical Notation	229
B.2.1	User and function specification	229
B.2.2	Flow of control specification	230
Appendix C.	Snart	232
C.1	Facets	232
C.2	Query Language	233
C.2.1	Introduction	233
C.2.2	Example schemas	234
C.2.3	Old style queries in Snart	235
C.2.4	New style queries in Snart	235
C.2.5	Implementation of the query language in Snart	239
C.3	Object Spaces	239
C.3.1	Introduction	240
C.3.2	Defining an object space in Snart	240
C.3.3	Working with an object space model in Snart	242

C.4 Persistency	243
C.4.1 Introduction	243
C.4.2 Persistency in the old Snart	244
C.4.3 Manipulating persistent objects in the old Snart	245
C.4.4 Persistency in the new Snart	246
C.5 ObjectViewer	248
Appendix D. Small Examples Schemas and Mappings	252
Appendix E. Large Example Schemas and Mappings	275
E.1 Description of Large Example	275
E.2 Schemas for the Large Example	278
E.2.1 IDM schema	278
E.2.2 PlanEntry schema	282
E.2.3 FaceEditor schema	286
E.2.4 VISION-3D schema	287
E.2.5 ThermalDesigner schema	289
E.3 Mappings for the Large Example	294
E.3.1 IDM <-> PlanEntry mapping	295
E.3.2 IDM <-> FaceEditor mapping	299
E.3.3 IDM <-> VISION-3D mapping	302
E.3.4 IDM <-> ThermalDesigner mapping	306
E.4 Project Window for the Large Example	309
E.4.1 User and function specification	309
E.4.2 Flow of control specification	313
Appendix F. The Parsers	320
F.1 ISO-10303:11 EXPRESS Parser	320
F.1.1 EXPRESS to Snart translator	322
F.1.2 Snart to EXPRESS translator	323
F.1.3 Snart to Reflex translator	323
F.2 ISO-10303:21 STEP data-file Parser	326
F.2.1 STEP data-file to Snart translator	328
F.2.2 Snart to STEP data-file translator	329
F.3 CGE Parser	329
F.4 VML Parser	331
Appendix G. Generalised Schema Representation Notation	332
G.1 Version Tree	332
G.2 Schema	333
Appendix H. Generalised Mapping Representation Notation	335
H.1 Schema	335
H.2 Mapping	336
H.3 Inverted Index	337
Glossary	338
References	342

List of Figures

Figure 1.1 COMBINE structure	5
Figure 1.2 Structure of an integrated design system	7
Figure 1.3 An example of the Smart graphical notation	12
Figure 1.4 A building described in PlanEntry	13
Figure 1.5 The PlanEntry schema main classes	14
Figure 1.6 Wall and window materials being defined in the FaceEditor	15
Figure 1.7 The FaceEditor schema main classes	15
Figure 1.8 A wire-frame representation of Figure 1.3 in VISION-3D	16
Figure 1.9 The VISION-3D schema classes	16
Figure 1.10 Control windows in ThermalDesigner	17
Figure 1.11 The IDM schema classes	18
Figure 2.1 EXPRESS and EXPRESS-G views in the EPE modelling environment	27
Figure 2.2 Example VML textual specification	32
Figure 2.3 Graphical mapping specification in VPE	33
Figure 2.4 Project flow of control definition	38
Figure 3.1 An example of the EXPRESS-G notation	44
Figure 3.2 Use of <i>technical_system</i> in the COMBINE IDM	45
Figure 3.3 Two high-level inheritance specifications for <i>technical_system</i>	46
Figure 3.4 Design stage specification of attributes of an entity	47
Figure 3.5 Textual view derived from graphical views of an entity	47
Figure 3.6 Constraining the cardinality of an attribute at late design stage	48
Figure 3.7 The propagation of an <i>update_record</i> to a dependant textual view	49
Figure 3.8 The automatic application of an update in a textual view	49
Figure 3.9 The persistent <i>update_record</i> viewer with documentation facility	50
Figure 3.10 A textual documentation view	51
Figure 3.11 The view navigator invoked for the <i>technical_system</i> entity	51
Figure 3.12 MViews three-layer multiple view architecture as used in SPE	54
Figure 3.13 Change propagation in an MViews environment	55
Figure 3.14 Class inheritance for EPE from MViews	56
Figure 5.1 A tree of <i>inter_view</i> mappings	85
Figure 5.2 Graphical mapping specification in VPE	99
Figure 5.3 Complex VML-G specification with textual equivalent	100
Figure 6.1 Initial connections between classes in two schemas	109
Figure 6.2 Specifying multiple mappings for a single class set	110
Figure 6.3 Defining links between all features associated with an <i>inter_class</i> definition	110
Figure 6.4 A full textual specification of an <i>inter_class</i> definition	111
Figure 6.5 Modification of a graphical view	112

Figure 6.6 Receipt of an <i>update_record</i> in a textual view	113
Figure 6.7 A textual view after manual application of an update specification	114
Figure 6.8 Browsing the change log for an <i>inter_class</i> specification	115
Figure 6.9 An associated documentation view for an <i>inter_class</i> definition	115
Figure 6.10 View navigation for <i>inter_class</i> specifications	116
Figure 6.11 Class inheritance for VPE from MViews	119
Figure 7.1 Multiple project windows in a project	127
Figure 7.2 Example of user and function specification	131
Figure 7.3 Invocable design functions	135
Figure 7.4 Constrained design function	135
Figure 7.5 Constraints between two design functions	136
Figure 7.6 Design function constraints leading to apparent inconsistencies	136
Figure 7.7 Top level flow of control specification	137
Figure 7.8 Flow of control with global elements	138
Figure 8.1 Sample mapping controllers	151
Figure 8.2 An operating flow of control manager	154
Figure 9.1 Instance viewing and navigation	158
Figure 9.2 InSTEP's graphical and textual views	159
Figure 9.3 A SmartQuery dialogue and result	161
Figure 9.4 A default ObjectViewer object layout	161
Figure 9.5 Reflex's multiple graphical views, along with an object's attribute dialogue	162
Figure 10.1 Structure of the traced persistent space in Smart	166
Figure 10.2 An actor's mapping controller interface	169
Figure 10.3 A transaction-based mapping manager	170
Figure 10.4 An automatic mapping manager	171
Figure 11.1 Calculating potential design functions from a CombiNet	192
Figure 11.2 Multiple actors causing select design functions to become stalled	193
Figure 11.3 Initial CombiNet in a project window	194
Figure 11.4 Multiple levels of aggregate functions	195
Figure 11.5 Exiting from a CombiNet representing an aggregate place	196
Figure 11.6 Design tool start up dialogue	197
Figure 11.7 Design tool termination dialogue	198
Figure 11.8 Project manager user interface	199
Figure 11.9 Project manager navigation through CombiNets	200
Figure 11.10 Actor's user interface	201
Figure A.1 VML graphical notation	219
Figure B.1 Icons used for user and function specification	229
Figure B.2 Icons used for flow of control specification	230
Figure C.1 Smart query language interface	234
Figure C.2 ObjectViewer interface	248
Figure E.1 Integration of tools in example	275
Figure E.2 Building design in PlanEntry with mapping controllers	276
Figure E.3 Result of mapping to VISION-3D and FaceEditor	276
Figure E.4 Building design after mapping to three tools	277
Figure E.5 Layout change to building in Figure E.4	277
Figure E.6 Result of propagating changes shown in Figure E.5	278
Figure E.7 IDM schema	279
Figure E.8 PlanEntry in use	283
Figure E.9 Planes calculated for a building	283
Figure E.10 PlanEntry schema	284
Figure E.11 FaceEditor in use	285

Figure E.12 FaceEditor schema	286
Figure E.13 VISION-3D in use	287
Figure E.14 VISION-3D schema	287
Figure E.15 ThermalDesigner in use	290
Figure E.16 Client and design roles	310
Figure E.17 Architect and design roles	311
Figure E.18 Structural consultant and design roles	311
Figure E.19 Daylighting consultant and design roles	312
Figure E.20 Thermal consultant and design roles	312
Figure E.21 Top-level CombiNet	313
Figure E.22 Design and update CombiNet	314
Figure E.23 Building design CombiNet	315
Figure E.24 Structural work CombiNet	316
Figure E.25 Daylight work CombiNet	317
Figure E.26 Thermal work CombiNet	318
Figure E.27 Acceptance CombiNet	319

List of Tables

Table 4.1 Mapping types from van Horssen et al. 1994	65
Table 4.2 Full set of structural mapping types	65
Table 4.3 Schema integration conflict types from Batini et al. 1986	66
Table 4.4 Schema integration conflict types from Kim and Seo 1991	67
Table 4.5 Schema fragments for the two schemas in the mapping example	70
Table 4.6 EXPRESS-M mapping for example problem	71
Table 4.7 EXPRESS-V mapping for example problem	73
Table 4.8 EXPRESS-C mapping for example problem	75
Table 4.9 Transformr mapping for example problem	76
Table 4.10 EDM-2 mapping for example problem	77
Table 4.11 KIF mapping for example problem	78
Table 4.12 Comparison of mapping languages	80
Table 5.1 VML mapping for example problem	83
Table 5.2 Top level definition of a VML mapping	83
Table 5.3 Definition of an <i>inter_view</i> specification	84
Table 5.4 Definition of an <i>inter_class</i> specification	86
Table 5.5 Definition of a <i>class_name</i>	88
Table 5.6 Definition of inheritance	89
Table 5.7 Definition of invariants	89
Table 5.8 Definition of initialisers	90
Table 5.9 Definition of equivalences	91
Table 7.1 Capabilities of project specification languages (after Curtis et al. 1992)	128
Table 10.1 Pseudo-code for performing a mapping	173
Table 10.2 Pseudo-code for establishing a mapping	173
Table 10.3 Pseudo-code for performing mappings	174
Table 10.4 Examples of <i>inter_class</i> headers and resultant object lists	176
Table 10.5 Pseudo-code for determining initial object groups	176
Table 10.6 Pseudo-code for generating object combinations for an <i>inter_class</i>	177
Table 10.7 Pseudo-code for the four-pass <i>inter_class</i> resolution	180
Table 10.8 Pseudo-code for mapping a new combination	181
Table 10.9 Pseudo-code for creating a new object	181
Table 10.10 Pseudo-code for mapping an object deletion	182
Table 10.11 Pseudo-code for mapping an object modification	183
Table 10.12 Pseudo-code for identifying values for an equation	185
Table 10.13 Update authorities for attributes derived from different sources	186
Table G.1 Specification of a schema version	332
Table G.2 Specification of version creation reasons	333
Table G.3 Specification of schema information	333
Table G.4 Specification of modification types	334

Table H.1 Specification of schema entities	335
Table H.2 Specification of atomic mapping components	336
Table H.3 Specification of inverted indices into mappings	337