

UNIVERSITE DE LIMOGES

ECOLE DOCTORALE Science – Technologie – Sante

Faculté de Science – Technologie – Sante

Année : 2005

Thèse N°

THESE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE LIMOGES

Discipline : Informatique

Présentée et soutenue publiquement par

Dimitrios MAKRIS

Le 18 Octobre 2005

*Etude et réalisation d'un système déclaratif de modélisation et de
génération de styles par algorithmes génétiques.
Application à la création architecturale*

Thèse dirigée par : Dimitri PLEMENOS Professeur

Coencadrement : Georges MIAOULIS Professeur - TEI d'Athènes

JURY :

Rapporteurs :

Véronique GAILDRAT Maitre de conférences HDR Université Paul Sabatier

Nikolaos VASSILAS Professeur Associe Institut d'Education Technologique d'Athènes

Examineurs :

Georges MIAOULIS Professeur Institut d'Education Technologique d'Athènes

Djamchid GHAZANFARPOUR Professeur Université de Limoges MSI

Dimitri PLEMENOS Professeur Université de Limoges MSI

Invitée :

Pascale FRIBAULT Architecte docteur Université de Limoges

Remerciements

Je souhaite remercier Madame Véronique Gaildrat, Maître de Conférences HDR à l'Université Paul Sabatier, pour avoir accepté d'être rapporteur, et pour ses remarques pertinentes et constructives concernant la rédaction de ce mémoire. Je remercie Monsieur Nikolaos Vassilas, Professeur Associé d'Informatique à l'Institut d'Education Technologique d'Athènes, pour avoir accepté d'être rapporteur, malgré une lourde charge de travail.

Je remercie Monsieur George Miaoulis, Professeur d'Informatique à l'Institut d'Education Technologique d'Athènes, pour avoir accepté de faire partie à mon jury. Je tiens à remercier Monsieur George Miaoulis pour avoir guidé mes études depuis de nombreuses années.

Je remercie Monsieur Djamchid Ghazanfarpour, Professeur d'Informatique à l'Université de Limoges et directeur du Laboratoire MSI, pour avoir accepté de faire partie à mon jury.

Je tiens à remercier Monsieur Dimitri Pléménos, Professeur d'Informatique à l'Université de Limoges, pour avoir guidé mes études depuis de nombreuses années, mais surtout avoir été mon directeur de thèse, tout en ayant su faire preuve de patience, d'écoute, d'ouverture et de beaucoup de disponibilité.

Je remercie également Madame Pascale Fribault architecte – docteur de l'Université de Limoges qui a accepté notre invitation pour faire partie du jury.

Je remercie aussi George Bardis, John Dragonas, Vasilis Golfopoulos et Ioanna Ravani pour leur collaboration. Je remercie également Ioannis Havoutis et Vasilis Stathopoulos pour leur contribution au logiciel.

ABSTRAIT

Dans ce mémoire nous étudions la possibilité d'introduire le concept de style dans la modélisation déclarative afin de faciliter la conception architecturale pendant la phase conceptuelle. Pour étudier une telle hypothèse, une méthodologie de la phase de conception architecturale utilisant le style a été développée. La méthode est placée dans le cadre de la modélisation déclarative et des algorithmes évolutionnaires.

En particulier nous développons le cadre de modélisation des connaissances architecturales. Nous proposons une modélisation du style dans le cycle de conception déclaratif et un moteur de génération de solutions basé sur un algorithme génétique multi-objectif.

Le résultat est le système prototype de conception déclarative évolutionnaire-MultiCAD d'aide à la conception architecturale. La méthode résultante permet de quantifier avec succès des critères qualitatifs esthétiques d'un bâtiment. Notre système constitue un outil de conception esthétique assisté par ordinateur.

Mots clés : modélisation déclarative, conception évolutionnaire, algorithme génétique multi-objectif, conception architecturale, style architecturale, modélisation des connaissances.

STUDY AND REALISATION OF A DECLARATIVE SYSTEM FOR
MODELLING AND GENERATION OF STYLE WITH GENETIC ALGORITHMS.
APPLICATION IN ARCHITECTURAL DESIGN

In this thesis we study the possibility to introduce the concept of style in declarative modelling in order to aid architectural design during the conceptual phase. In order to study such a hypothesis we have develop a methodology of architectural conceptual design utilising style. The method is based in the frame of declarative modelling, and evolutionary algorithms.

In particular, we develop a framework for the modelling of architectural knowledge. We define a model of style based on the declarative conceptual cycle. Moreover we develop a new engine for the generation of solutions. This engine is based on multi-objective genetic algorithms.

The result is a system prototype for Evolutionary Declarative Design-MultiCAD for the aid of conceptual architectural design. The resulted method could successfully quantify aesthetic qualitative criteria of a building. Our system points towards a computer aided aesthetic design tool.

Keywords: Declarative modelling, evolutionary design, multi-objective genetic algorithms, architectural design, architectural style, knowledge modelling

Table of contents

| | |
|--|----|
| Chapitre 1 Introduction | 1 |
| 1.1. Introduction..... | 1 |
| 1.1.1. Conception Architecturale | 2 |
| 1.1.2. Style | 3 |
| 1.1.3. Modélisation Déclarative | 4 |
| 1.1.4. Conception évolutionnaire | 5 |
| 1.2. Objectifs de recherche..... | 7 |
| 1.3. Approche et méthodologie de recherche..... | 8 |
| 1.4. Signification - Avantages Potentiels | 9 |
| 1.5. Grandes lignes de la thèse..... | 10 |
| Chapter 2 General Review of Literature | 15 |
| 2.1. Introduction..... | 15 |
| 2.2. Declarative modelling | 16 |
| 2.2.1. Advantages of declarative modelling..... | 17 |
| 2.2.2. Limitations of declarative modelling | 18 |
| 2.2.3. Declarative modellers | 18 |
| 2.2.4. General and dedicated declarative modellers | 20 |
| 2.2.5. Declarative Modelling by Hierarchical Decomposition | 21 |
| 2.2.5.1. DMHD modellers..... | 22 |
| 2.2.5.1.1. MultiFormes..... | 22 |
| 2.2.5.1.2. Declarative Modelling of Habitation Edifices | 23 |
| 2.2.6. MultiCAD II..... | 23 |
| 2.2.6.1. MULTICAD-II Framework | 24 |
| 2.2.6.1.1. MultiCAD-II Software architecture | 24 |
| 2.2.7. Discussion | 27 |
| 2.2.7.1. Description phase..... | 28 |
| 2.2.7.2. Generation-understanding phase..... | 28 |
| 2.3. Evolutionary Search..... | 30 |
| 2.3.1. Genetic Algorithms | 32 |
| 2.3.1.1. Chromosomal representation | 34 |
| 2.3.1.2. Initialising population | 34 |
| 2.3.1.3. Evaluation function..... | 35 |

| | |
|---|----|
| 2.3.1.4. Selection methods | 35 |
| 2.3.1.4.1. Fitness proportionate selection | 35 |
| 2.3.1.4.2. Rank selection | 37 |
| 2.3.1.5. Recombination operations | 38 |
| 2.3.1.5.1. Crossover | 38 |
| 2.3.1.5.2. Mutation | 39 |
| 2.3.1.5.3. Inversion | 39 |
| 2.3.1.6. Population size | 40 |
| 2.3.1.7. Advanced Genetic Algorithms | 40 |
| 2.3.2. Multi-objective Optimisation | 41 |
| 2.3.2.1. A posteriori methods | 42 |
| 2.3.2.2. Progressive methods | 42 |
| 2.3.2.3. A priori methods | 42 |
| 2.3.3. Handle constraints in Evolutionary Algorithms | 43 |
| 2.3.3.1. Death penalty | 43 |
| 2.3.3.2. Penalisation methods | 44 |
| 2.3.3.3. Decoders | 44 |
| 2.3.3.4. Repair strategy | 44 |
| 2.3.3.5. Constraint-preserving operators | 44 |
| 2.3.3.6. Separation of Objectives and constraints | 44 |
| 2.3.4. Choice of weights | 44 |
| 2.3.5. Evolutionary Design | 45 |
| 2.3.6. Genetic algorithm applications in design | 45 |
| 2.3.6.1. Conceptual design | 46 |
| 2.3.6.2. Shape optimisation | 49 |
| 2.3.7. Discussion | 49 |
| 2.3.7.1. Why Evolutionary and Adaptive Computing? | 50 |
| 2.3.7.2. Justification of Choice Using Genetic Algorithms | 51 |
| 2.4. Architectural Design | 52 |
| 2.4.1. Hard – Soft constraints | 54 |
| 2.4.2. Architectural building design phases | 56 |
| 2.4.2.1. Conceptual design | 56 |
| 2.4.2.2. Preliminary design | 57 |
| 2.4.2.3. Embodiment design | 57 |

| | |
|---|----|
| 2.4.2.4. Detail design | 57 |
| 2.4.2.5. Problems in conceptual design..... | 58 |
| 2.4.3. Architectural Design knowledge modelling | 59 |
| 2.4.3.1. Building modelling | 60 |
| 2.4.3.2. Architectural knowledge in CAD systems..... | 63 |
| 2.4.4. Feature-based Modelling | 64 |
| 2.4.4.1. Features Categories | 65 |
| 2.4.4.2. Feature-based modelling approach | 66 |
| Chapter 3 Architectural Style..... | 69 |
| 3.1. Introduction..... | 69 |
| 3.2. The concept of style | 70 |
| 3.2.1. The object view | 70 |
| 3.2.2. The process view..... | 71 |
| 3.3. Identification of style | 72 |
| 3.3.1. Measurement of style | 73 |
| 3.3.2. Degree of style | 74 |
| 3.3.3. Style in Architectural design..... | 74 |
| 3.3.4. Architectural style | 75 |
| 3.3.5. Morphological features | 76 |
| 3.3.6. Semantic features | 79 |
| 3.4. Modelling architectural style | 81 |
| 3.4.1. Stylistic Features | 82 |
| 3.4.1.1. Spatial Features | 82 |
| 3.4.1.2. Structural Features | 83 |
| 3.4.2. Stylistic Principles | 83 |
| 3.4.2.1. Topological Principles | 83 |
| 3.4.2.2. Formative Principles | 84 |
| 3.5. Two case studies of architectural style | 86 |
| 3.5.1.1. Style “Santorini” | 86 |
| 3.5.1.2. Style “Metsovo” | 88 |
| 3.6. Computer-supported approaches for aesthetic design | 90 |
| 3.7. Discussion | 93 |
| Chapter 4 Research Statement | 95 |
| 4.1. Introduction..... | 95 |

| | |
|---|-----|
| 4.2. Architectural knowledge in MultiCAD | 96 |
| 4.2.1. Architectural knowledge framework for building modelling | 97 |
| 4.2.2. Framework analysis | 98 |
| 4.2.3. Architectural Constraints | 101 |
| 4.2.4. Knowledge representation in MultiCAD | 103 |
| 4.2.5. Normalised Declarative Building Model | 104 |
| 4.3. Architectural style in MultiCAD | 106 |
| 4.3.1. Modelling scheme | 107 |
| 4.3.2. Style Knowledge in MultiCAD software architecture | 107 |
| 4.3.2.1. Style framework | 108 |
| 4.3.2.2. Style representational scheme | 108 |
| 4.3.2.3. Defining semantics for architectural style | 109 |
| 4.3.2.4. Stylistic constraints - properties | 110 |
| 4.3.2.5. Definition of objects | 110 |
| 4.3.3. Framework analysis | 111 |
| 4.3.3.1. Spatial planning constraints | 112 |
| 4.3.3.1.1. Hard constraints | 112 |
| 4.3.3.1.2. Soft constraints | 113 |
| 4.3.3.2. Aesthetic evaluation | 114 |
| 4.3.3.3. Roof morphology constraints | 115 |
| 4.3.3.3.1. Hard constraints | 115 |
| 4.3.3.3.2. Soft constraints | 116 |
| 4.3.3.4. Measure of Style | 116 |
| 4.3.4. Style in Description phase | 117 |
| 4.3.5. Style in Generation phase | 118 |
| 4.4. Geometric level of representation | 118 |
| 4.4.1. Spatial object geometry | 118 |
| 4.4.2. Roof object geometry | 119 |
| 4.5. Simple Genetic algorithm | 121 |
| 4.5.1. Phenotype | 122 |
| 4.5.2. Genotype Chromosome Encoding | 123 |
| 4.5.3. Evaluation functions | 124 |
| 4.5.4. Selection mechanism | 124 |
| 4.5.5. Genetic operators | 124 |

| | |
|---|-----|
| 4.5.6. Reproducing new generations..... | 125 |
| 4.6. Multi-objective genetic algorithm..... | 125 |
| 4.6.1. MOGA system | 125 |
| 4.6.1.1. Phenotype..... | 125 |
| 4.6.1.2. Genotype | 126 |
| 4.6.1.2.1. Spatial chromosome..... | 127 |
| 4.6.1.2.2. Structural chromosome | 127 |
| 4.6.1.3. Genetic Algorithm | 128 |
| 4.6.1.4. Evaluation functions | 129 |
| 4.6.1.4.1. Multi-objective functions..... | 130 |
| 4.6.1.5. Genetic operators - Crossover..... | 131 |
| 4.6.1.6. Mutation..... | 131 |
| 4.6.1.7. Selection..... | 132 |
| 4.6.1.8. Initialisation | 132 |
| 4.6.2. Implementation of Evolutionary Design Environment..... | 132 |
| Chapter 5 Evolutionary Declarative Design Prototype System..... | 135 |
| 5.1. Introduction..... | 135 |
| 5.2. DKABM mechanism implementation | 135 |
| 5.2.1. Components of DKBM implementation..... | 137 |
| 5.2.1.1. Interface layer | 137 |
| 5.2.1.2. Process layer | 137 |
| 5.2.1.3. Database Management layer | 138 |
| 5.2.2. Using DKBM in Design cases | 138 |
| 5.2.2.1. Interfaces..... | 138 |
| 5.2.3. Discussion..... | 142 |
| 5.3. MultiCAD – GA | 143 |
| 5.3.1. Basic implementation structure..... | 143 |
| 5.3.2. Implementation and simulations | 144 |
| 5.4. MultiCAD – MOGA | 146 |
| 5.4.1. Functional aspects of methodology | 146 |
| 5.4.2. Software architecture | 148 |
| 5.4.2.1. User Interface Layer | 149 |
| 5.4.2.2. Processing Layer..... | 150 |
| 5.4.2.3. Developers Features..... | 150 |

| | |
|---|-----|
| 5.4.2.4. Data Management Layer | 151 |
| 5.4.3. Functional models | 152 |
| 5.4.3.1. Description phase – Model Description (Activity A1) | 153 |
| 5.4.3.2. Description phase – Fitness Function Selection (Activity A1) | 154 |
| 5.4.3.3. Solutions Generation (Activity A2) | 155 |
| 5.4.3.4. Visualisation Understanding (Activity A3) | 156 |
| 5.4.4. Components Interface | 157 |
| 5.4.4.1. Model Description Interfaces (Activity A1) | 157 |
| 5.4.4.2. Fitness Function Selection Interfaces (Activity A1) | 158 |
| 5.4.4.3. Solutions Generation Interfaces (Activity A2) | 159 |
| 5.4.4.4. Visualization-Understanding Interface | 160 |
| 5.5. Discussion | 161 |
| Chapter 6 Experiments – System Evaluation | 163 |
| 6.1. Introduction | 163 |
| 6.2. Santorini Style | 163 |
| 6.2.1. Space Planning | 164 |
| 6.2.1.1. Scene A | 164 |
| 6.2.1.1.1. Definition of Scene | 164 |
| 6.2.1.1.2. Room and Building Objective Functions | 166 |
| 6.2.1.1.3. Experiment – A | 167 |
| 6.2.1.2. Scene B | 170 |
| 6.2.1.2.1. Experiment B | 170 |
| 6.2.2. Roof Morphology | 174 |
| 6.2.2.1. Scene A | 175 |
| 6.2.2.1.1. Definition of Roof(s) | 175 |
| 6.2.2.1.2. Roof Objective Functions | 176 |
| 6.2.2.1.3. Experiment A | 177 |
| 6.2.2.2. Scene B | 181 |
| 6.2.2.2.1. Definition of Roof(s) | 181 |
| 6.2.2.2.2. Roof Objective Functions | 182 |
| 6.2.2.2.3. Experiment B | 182 |
| 6.3. Metsovo Style | 185 |
| 6.3.1. Space planning | 185 |
| 6.3.1.1. Scene A | 185 |

| | |
|---|-----|
| 6.3.1.1.1. Definition of Scene | 185 |
| 6.3.1.1.2. Room and Building Objective Functions..... | 186 |
| 6.3.1.1.3. Experiment A | 187 |
| 6.3.1.2. Scene B | 190 |
| 6.3.1.2.1. Definition of Scene | 190 |
| 6.3.1.2.2. Room and Building Objective Functions..... | 191 |
| 6.3.1.2.3. Experiment B | 192 |
| 6.3.2. Roof Morphology..... | 195 |
| 6.3.2.1. Scene A | 195 |
| 6.3.2.1.1. Definition of Roof(s)..... | 195 |
| 6.3.2.1.2. Experiment A | 196 |
| 6.3.2.2. Scene B | 199 |
| 6.3.2.2.1. Definition of Roof(s)..... | 199 |
| 6.3.2.2.2. Roof Objective Functions | 200 |
| 6.3.2.2.3. Experiment B | 201 |
| 6.4. Discussion | 203 |
| Chapitre 7 Discussion - Conclusions | 207 |
| 7.1. Discussion | 207 |
| 7.2. Conclusion | 211 |
| 7.2.1. Modélisation déclarative et phase de conception architecturale..... | 211 |
| 7.2.2. Esthétique et intelligence artificielle..... | 212 |
| 7.3. Travaux futurs - Questions ouvertes | 212 |
| 7.3.1. Expérimentation du système de prototype..... | 212 |
| 7.3.2. Extensibilité du processus d'évaluation | 213 |
| 7.3.3. Extensibilité à d'autres domaines de conception..... | 213 |
| 7.3.4. Application - Utilisation en tant qu'outil d'enseignement de conception | 213 |
| Index of Abbreviation | 214 |
| Bibliography | 216 |
| Appendix..... | 235 |

Table of Figures

| | |
|--|-----|
| Figure 2-1 MultiCAD-II layered architecture..... | 25 |
| Figure 3-1 Byzantine style..... | 77 |
| Figure 3-2 Greek temple..... | 78 |
| Figure 3-3 Byzantine – Gothic style..... | 78 |
| Figure 3-4 The generation of style from design decisions..... | 79 |
| Figure 3-5 Metsovo house style..... | 80 |
| Figure 3-6 Santorini habitation..... | 88 |
| Figure 3-7 Metsovo habitation..... | 90 |
| Figure 4-1 Scene Meta-model and Knowledge description in MultiCAD..... | 97 |
| Figure 4-2 DKABM: Entries with aggregation-generalisation relations..... | 99 |
| Figure 4-3 Obligatory (highlighted)-Permissive relations (simple arrows) between entities in DKABM..... | 101 |
| Figure 4-4 Scene and Knowledge <i>Meta-model</i> in MultiCAD..... | 103 |
| Figure 4-5 Representation of ‘ <i>MY_House</i> ’..... | 105 |
| Figure 4-6 Normalized Declarative Building Model of ‘ <i>MY_House</i> ’ habitation..... | 106 |
| Figure 4-7 Isothetic box..... | 118 |
| Figure 4-8 Frames A-B-C-D-E-F-G-H..... | 119 |
| Figure 4-9 Fitting points in a frame..... | 119 |
| Figure 4-10 Frames a-b-c-d..... | 120 |
| Figure 4-11 $D_{OFFSET} Length - D_{OFFSET} Width$ | 120 |
| Figure 4-12 $O_{OFFSET} Length - O_{OFFSET} Width$ | 120 |
| Figure 4-13 Examples of roof forms..... | 121 |
| Figure 4-14 Examples of basic volumes..... | 122 |
| Figure 4-15 Polyface mesh..... | 123 |
| Figure 4-16 Chromosome representation of a scene with two subscenes..... | 123 |
| Figure 5-1 MultiCAD II Software architecture in a commercial CAD system..... | 136 |
| Figure 5-2 a) Interface for the description of normalized scenes, b) An example of normalized description of Santorini’s style home..... | 139 |
| Figure 5-3 a) A scene solution of Santorini house and the respective Scene Base records. b) Interface for the data management of scenes..... | 140 |
| Figure 5-4 Interface of tree view representation of scene solutions..... | 140 |
| Figure 5-5 Examples of scene solutions..... | 141 |

| | |
|---|-----|
| Figure 5-6 MultiCAD-GA system organisation | 143 |
| Figure 5-7 MultiCAD-GA Functions & Data structures | 144 |
| Figure 5-8 Two solutions for the residence scene | 145 |
| Figure 5-9 Declarative evolutionary design process..... | 146 |
| Figure 5-10 Flowchart of applied methodology | 147 |
| Figure 5-11 MultiCAD MOGA Implementation Architecture | 148 |
| Figure 5-12 Manual score input for automatic weight calculation | 151 |
| Figure 5-13 Scene base | 152 |
| Figure 5-14 Idef Level 0 | 153 |
| Figure 5-15 Model Description Activity..... | 154 |
| Figure 5-16 Fitness Function Selection Activity | 155 |
| Figure 5-17 Solutions Generation Activity..... | 156 |
| Figure 5-18 Solutions Visualisation Activity | 156 |
| Figure 5-19 <i>Model Information & Model Properties</i> interface | 157 |
| Figure 5-20 <i>Model Constraints & Roofs Relations</i> interface..... | 158 |
| Figure 5-21 <i>Fitness Functions Definition</i> interface..... | 158 |
| Figure 5-22 <i>Solutions Generation</i> interface..... | 159 |
| Figure 5-23 Graphic visualisation interface..... | 160 |
| Figure 6-1 Model information | 165 |
| Figure 6-2 Room properties | 165 |
| Figure 6-3 Room constraints..... | 166 |
| Figure 6-4 Rooms fitness functions – Building fitness functions..... | 166 |
| Figure 6-5 Genetic Algorithm Information..... | 167 |
| Figure 6-6 Best individual from generation 649 – Plan view | 170 |
| Figure 6-7 Best individual from generation 649 – South west view | 170 |
| Figure 6-8 Genetic Algorithm Information..... | 171 |
| Figure 6-9 Best individual from generation 969 – Plan view | 172 |
| Figure 6-10 Best individual from generation 969 – South west view | 172 |
| Figure 6-11 Genetic Algorithm Information..... | 173 |
| Figure 6-12 Best individual from generation 729 – Plan view | 174 |
| Figure 6-13 Best individual from generation 729 – South west view | 174 |
| Figure 6-14 Roofs relations | 176 |
| Figure 6-15 Roof Fitness Functions..... | 177 |
| Figure 6-16 Genetic Algorithm Information..... | 178 |

| | |
|--|-----|
| Figure 6-17 Best object from loop 2, generation 659. Front side view | 180 |
| Figure 6-18 Best object from loop 2, generation 659. Rear side view | 180 |
| Figure 6-19 Best object from loop 2, generation 659. Rear side view | 180 |
| Figure 6-20 Room – Roof Relations | 182 |
| Figure 6-21 Roofs Fitness Function..... | 182 |
| Figure 6-22 Genetic Algorithm Information..... | 183 |
| Figure 6-23 Best object from loop 2, generation 359. Front side view | 184 |
| Figure 6-24 Best object from loop 2, generation 539. Rear side view | 184 |
| Figure 6-25 Model Information | 186 |
| Figure 6-26 Room Properties..... | 186 |
| Figure 6-27 Rooms Constraints | 186 |
| Figure 6-28 Building – Room Fitness functions..... | 186 |
| Figure 6-29 Genetic Algorithm Information..... | 187 |
| Figure 6-30 Best individual from generation 479 – Plan view | 190 |
| Figure 6-31 Best individual from generation 479 – South west view | 190 |
| Figure 6-32 Model Information | 191 |
| Figure 6-33 Rooms Properties | 191 |
| Figure 6-34 Rooms Constraints | 191 |
| Figure 6-35 Building – Rooms Fitness Function..... | 191 |
| Figure 6-36 Genetic Algorithm Information..... | 192 |
| Figure 6-37 Best individual - generation 829 – Plan view | 194 |
| Figure 6-38 Best individual - generation 829 – South west view..... | 194 |
| Figure 6-39 Roofs – Rooms Relations..... | 196 |
| Figure 6-40 Roof Fitness Functions..... | 196 |
| Figure 6-41 Genetic Algorithm Information..... | 197 |
| Figure 6-42 Best object from loop 2 and generation 679. South view | 199 |
| Figure 6-43 Best object from loop 2 and generation 679. Southwest view | 199 |
| Figure 6-44 Roofs – Rooms relations | 200 |
| Figure 6-45 Roofs Fitness Functions | 201 |
| Figure 6-46 Genetic Algorithm Information..... | 201 |
| Figure 6-47 Best object from loop 2 and generation 599. South front view | 203 |
| Figure 6-48 Best object from loop 2 and generation 599. Southwest view..... | 203 |

List of Tables

| | |
|--|-----|
| Table 3-1 Santorini style | 87 |
| Table 3-2 Metsovo Style | 89 |
| Table 4-1 Types of Properties and Relations in DKABM | 100 |
| Table 4-2 Definition of Living Room described in ‘ <i>MY_House</i> ’ building..... | 102 |
| Table 4-3 Spatial Chromosome | 127 |
| Table 4-4 Roof Chromosome..... | 128 |
| Table 6-1 Spatial planning examples from different generations..... | 169 |
| Table 6-2 Number of Roofs | 175 |
| Table 6-3 Room placement - dimensions | 176 |
| Table 6-4 Roof examples from different generations | 179 |
| Table 6-5 Number of Roofs | 181 |
| Table 6-6 Rooms placement - dimensions..... | 181 |
| Table 6-7 Spatial planning examples from different generations..... | 189 |
| Table 6-8 Number of Roofs | 195 |
| Table 6-9 Rooms placement – dimensions | 195 |
| Table 6-10 Roof designs along generations..... | 198 |
| Table 6-11 Number of Roofs | 199 |
| Table 6-12 Rooms placement – dimensions | 200 |



Chapitre 1

Introduction

1.1. Introduction

Les problèmes de conception architecturale n'appartiennent pas à des secteurs dans lesquels les problèmes pourraient être résolus par la combinaison d'une grande quantité de manipulations logiques et formelles et d'une quantité minimale de connaissances du contexte du monde réel. Ici, les solutions ne sont pas ce qu'elles sont dans d'autres domaines; Ce ne sont pas des solutions correctes, conditionnellement catégoriques et ainsi, capables d'être remplacées ou annulées par modification dans l'agrégat des conditions de support. Ce sont des solutions dans la mesure où elles nous permettent de penser qu'elles ne peuvent pas être réduites à un agrégat défini des conditions. La fonction des buts de conception est de motiver et d'inspirer l'activité qui à son tour produira de nouveaux buts [Simon 96]. Cela se produit parce que la nature de ces problèmes implique un comportement coopératif évolutif et/ou un paysage changeant et bien sûr, nous ne pouvons pas appliquer uniquement un optimiseur de fonction, quelle que soit la puissance de ses performances. On accepte [Tzonis 94] que le processus de la conception architecturale est basé sur des problèmes mal définis et c'est loin d'être un processus de routine. En fait, son inexactitude conduit le progrès vers la pensée architecturale. Pour cette raison, ces processus ne sont pas bien compris, et par conséquent, ne peuvent être simulés par aucune approche algorithmique simple. La génération de concepts dont la représentation est de forme et de taille_inconnue est une tâche très difficile. La création de formes spatiales est un exemple de ce type de concept. La synthèse - composition de la structure spatiale tridimensionnelle d'un espace à bâtir est l'une des tâches ouvertes les plus importantes dans l'architecture. Les architectes travaillent de manière à impliquer la formulation de beaucoup d'articles d'information alors que ces articles sont d'importance variable pour les conceptions. Ils identifient leur applicabilité, tout en travaillant les détails de la solution. Pendant le processus de conception, il n'est pas possible de commencer par des listes de critères auxquels la

disposition nouvellement créée est censée répondre. Tout le processus pour une bonne solution est également de rechercher une information appropriée avec laquelle l'évaluer [Rychener 88].

La phase conceptuelle du processus de conception architecturale, dans un contexte donné, est aussi dynamique que la recherche des rapports naissants entre les conceptions, leur style et le contexte lui-même. De cette façon, le style agit tel un principe d'ordre dans la conception qui permet aux objets et aux processus de construction d'être structurés, apportant l'ordre à un domaine qui autrement serait apparemment chaotique.

1.1.1. Conception Architecturale

Dans l'architecture, le procédé de « résolution de problèmes » concerne des activités de conception. Ces activités pourraient être caractérisées par leur forte créativité. Dans la conception architecturale, il n'existe aucune stratégie prédéterminée pour réaliser un ensemble de buts. En général, le processus de conception est caractérisé par sa complexité, son imprécision et son degré élevé de subjectivité [Simon 96].

Nous pourrions définir la conception architecturale comme: *le processus consistant à appliquer diverses techniques et principes afin de définir un système spatial avec suffisamment de détails pour permettre sa réalisation physique. Pour n'importe quel produit ou système architectural, la toute première étape de la phase de développement est la conception dans le but de produire un modèle ou une représentation d'une entité qui sera construite par la suite. Le processus agit en combinant l'intuition et le jugement basé sur l'expérience construisant des modèles semblables, un ensemble de principes qui guident le modèle dans son évolution, un ensemble de critères qui permet à la qualité d'être jugée et un processus de répétition que mène à une représentation de conception finale.*

Il est très important qu'il émerge une compréhension des conditions émerge pendant que l'activité de conception se poursuit; c'est pourquoi les conditions de conception sont généralement « mal définies » et souvent contradictoires [Simon 96]. La conception est un processus dynamique. Les méthodes compositionnelles incorporent la transformation de métaphores familières et les projettent dans de nouvelles situations. La connaissance de la conception pourrait être analysée sous deux catégories. La première catégorie concerne la connaissance rationnelle qui inclut des codes de conception et l'analyse de la fonction. La deuxième catégorie concerne la

composition spatiale, la planification de l'espace et le style architectural [Rowe 91] [Coyne 90].

Pendant le processus de conception, les architectes basent principalement leur recherche sur la connaissance qui définit un modèle architectural. Tandis que le style existant est combiné et rassemblé, il y a un contrôle et une optimisation des demandes fonctionnelles données [Smithies 81].

1.1.2. Style

Le style est très important dans la formation d'une conception de bâtiments. Il caractérise fortement un bâtiment puisqu'il permet au concepteur de transformer en groupant ou en classifiant les éléments de conception de bâtiments existants selon certaines propriétés et modèles reconnaissables [Gombrich 68], [Simon 75], [Meyer 70]. Le contexte ou la situation influence les processus de transformation qui peuvent se produire différenciellement par des actes d'imitation ou radicalement par des actes d'innovation [Shapiro 61], [Ackerman 63], [Smithies 1981]. Le style architectural agit en tant que règle ordonnatrice dans la conception architecturale qui permet aux éléments, aux modèles et aux processus de construction d'être structurés, fournissant un ordre [Beardsley 66], [Michelis 01], [Michelis 02].

Dans l'histoire de la conception architecturale, le style aide à la description des uniformités parmi des projets de bâtiments comme des créations d'un architecte particulier, d'une Ecole, d'une période culturelle ou d'une région géographique [Scruton 79], [Kruft 94]. Les études sur les uniformités ou les modèles de bâtiments et les dispositifs spatiaux sont une approche commune pour étudier et modeler l'analyse - critique architecturale. Le rôle du style pendant la phase de conception est cruciale parce que beaucoup d'intentions stylistiques fondamentales sont introduites par les concepteurs afin de guider le développement d'une conception [Simon 75].

L'introduction du style dans un système de *Conception Architecturale Assistée par Ordinateur (CAAO)* sera favorable au processus de conception architecturale. L'objectif principal de cette thèse est la modélisation du style et son utilisation dans les premières phases de la conception architecturale. Afin de réaliser ses intentions, cette thèse développera un système CAAO avec l'implication de méthodes et de techniques de Modélisation Déclarative et de Conception Evolutionnaire.

1.1.3. Modélisation Déclarative

Le but de Modélisation Déclarative est de remédier aux inconvénients de la modélisation géométrique classique en offrant la possibilité de décrire la scène en utilisant des propriétés, qui peuvent être précises ou imprécises. Plus justement, la modélisation déclarative permet à l'utilisateur de dire quelles sont les propriétés qui doivent vérifier une scène, sans indiquer la manière d'obtenir une scène avec ces propriétés [Lucas 89], [Lucas 95]. Puisque le concepteur de la scène n'a pas nécessairement une connaissance complète de tous les détails de la scène qu'il veut obtenir, il semble normal de lui permettre d'employer les propriétés imprécises pour décrire la scène.

Afin de permettre des descriptions à des niveaux de détail variés, une nouvelle technique de modélisation déclarative, nommée *Modélisation Déclarative par Décomposition Hiérarchique (MDDH)*, est ici présentée [Plemenos 91], [Plemenos 95]. Cette technique de modélisation emploie la description hiérarchique descendante (top-down) et fonctionne comme suit: si une scène est facile à décrire, elle est décrite par un nombre restreint de propriétés qui peuvent être des propriétés de taille soit « l'inter - dimension » (plus haut que large, aussi haut que profond, etc.) ou des propriétés de forme (allongé, très arrondi, etc.). Sinon, la scène est partiellement décrite avec des propriétés faciles à décrire et est alors décomposée en un nombre de sous - scènes et le même processus de description est appliqué à chaque sous - scène. Afin d'exprimer les rapports dans les sous - scènes d'une scène, on utilise les propriétés de placement (mettre dessus, coller sur le côté gauche, etc.) et les propriétés de taille, soit « les inter-scènes », (plus haut que, etc.). Ce modèleur déclaratif permet de décrire des scènes de la façon hiérarchique et il peut engendrer toutes les scènes possibles correspondant à la description. La production de scènes permet d'obtenir des scènes à des niveaux de détails variés et même de mélanger les représentations approximatives et détaillées pour les différentes parties d'une scène.

Les systèmes de Conception Assistée par Ordinateur communs (CAO) fournissent en général un environnement typique de conception où le concepteur utilise certaines méthodes nécessaires dans un processus de conception ascendante (bottom-up). Nous considérons que les conceptions de construction architecturale sont complexes et en général peu connues puisque le concepteur y pense d'une manière plus abstraite.

La modélisation déclarative pourrait enrichir l'aspect dynamique et déclaratif d'une spécification de conception. La modélisation déclarative est une approche totale du processus de conception. La modélisation déclarative de la scène permet à la description des scènes d'être conçue en ne donnant que quelques propriétés requises de la scène et en laissant le modelleur trouver les solutions alternatives, le cas échéant, en vérifiant ces propriétés. Afin de modéliser des scènes complexes, on emploie préalablement la *Modélisation Déclarative par Décomposition Hiérarchique (MDDH)*, qui est basée sur conception descendante (top-down) des scènes.

La modélisation déclarative de scène est basée sur le cycle déclaratif de conception la description déclarative de scènes, la génération de solutions et la compréhension de scènes [Lucas 95], [Desmontils 95], [Colin 97].

Au début de la phase de la conception architecturale le dossier d'un bâtiment fournit seulement un assortiment de propriétés et une combinaison des contraintes rigides et légères. En général, beaucoup de détails numériques sont absents et apparaissent à l'émergence de l'objet de conception. Toutefois le concepteur n'a pas, dans plusieurs cas, une connaissance complète des objets à construire dès le commencement. La conception architecturale peut vraiment émerger avec la participation active de l'esprit du concepteur dans le processus déclaratif. Le paradigme de la conception de modélisation déclarative est une méthode appropriée qui pourrait cerner la nature de la conception et l'imprécision des objets de conception. La modélisation déclarative traite de la description imprécise des objets et offre au concepteur un environnement familier pour l'expression précise de l'idée de conception élaborée [Plemenos 02]. L'introduction des modèles déclaratifs spécifiques enrichirait le début de la phase de la conception architecturale. La modélisation déclarative a cette direction pour but.

1.1.4. Conception évolutionnaire

Des théories et des méthodologies ont été développées en l'*Intelligence Artificielle (IA)* et la conception évolutionnaire pour la génération automatique des concepts de construction et la recherche intelligente pour les alternatives de conception [Bentley 99]. On mentionne les théories et les méthodologies comme des techniques d'évaluation de conception génératives et évolutionnaires. Elles peuvent être employées par des concepteurs dans les secteurs de la conception architecturale et d'ingénierie pour synthétiser des concepts de construction initiaux à partir de conditions de conception fonctionnelle. Elles peuvent également être employées pour

l'exploration et l'optimisation des concepts initiaux afin de produire des solutions de conception alternatives. L'avantage principal tiré de l'emploi de ces techniques est de permettre aux concepteurs de se concentrer sur les aspects plus créatifs de la conception alors qu'il utilise la puissance d'évaluation massive pour les solutions pour produire et faire évoluer les solutions sollicitées [Goldberg 98], [Holland 92].

Nous décidons d'adopter l'évaluation évolutionnaire et adaptative pour les raisons suivantes: les techniques évolutionnaires et adaptatives peuvent présenter des avantages sur les techniques d'optimisation plus traditionnelles. Les attributs communs des diverses techniques de recherche évolutives et adaptatives d'importance particulière pour la résolution des problèmes pratique et complexe incluent :

- La condition de peu de connaissance a priori relative au problème, le cas échéant.
- Les capacités exploratoires. Les techniques produisent, au départ et de manière aléatoire, des solutions d'essai et l'ampleur de la recherche qui suivra à partir de telles solutions dépendra de leur rendement relatif.
- La capacité d'éviter des optimums locaux. La nature stochastique des divers algorithmes combinés avec l'échantillonnage aléatoire continu de l'espace de recherche peut empêcher la convergence vers des sous - optimums locaux.
- La capacité de manipuler des dimensions élevées. Une application réussie à des problèmes décrits par plus de quatre cents paramètres variables est possible.
- La robustesse à travers un large éventail de classes de problèmes. Les techniques peuvent généralement surpasser des algorithmes d'optimisation plus déterministes à travers un éventail de classes de problèmes.
- La proposition de plusieurs bonnes solutions. On peut développer des stratégies de recherche évolutionnaire et adaptatives qui identifient les solutions à rendement élevé et multiple.
- L'introduction des approches multi-objectives, qui peuvent être intégrées facilement et avec succès pour fournir une gamme de solutions de compromis à rendement élevé pour davantage d'évaluation en dehors du processus de production.

De telles techniques incluent des stratégies d'évolutionnaire ; algorithmes génétiques, programmation évolutionnaire, programmation génétique et modèles de colonie de fourmis, recherche de tabu et recuit simulé. Ces techniques sont fermement établies et sont maintenant largement utilisées dans l'industrie. Un *Algorithme Génétique (AG)*

est une technique de recherche adéquate pour trouver des solutions pour les espaces bruyants avec des minimums locaux et globaux. Parce qu'il recherche à partir d'une population de points, et non à partir d'un point unique, la probabilité que la recherche soit piégée dans un minimum local est faible. Les algorithmes génétiques amorcent la recherche par un échantillonnage aléatoire dans l'espace de solutions imprécises et par la suite utilisent les opérateurs stochastiques pour diriger un processus d'ascension basé sur des valeurs de fonction objective [Goldberg 89].

1.2. Objectifs de recherche

L'hypothèse principale de la thèse est la suivante : est-il possible d'introduire la notion de style dans le cadre de la modélisation déclarative afin de faciliter la conception architecturale pendant la phase conceptuelle ?

Afin d'étudier une telle hypothèse, nous développerons une méthodologie de conception architecturale utilisant le style. La méthode serait placée dans le cadre de la modélisation déclarative et des algorithmes évolutionnaires.

Une telle méthodologie implique le développement d'un système prototype de conception déclarative, et évolutionnaire pour l'apparition des compositions de bâtiments. Elle commence par la description déclarative des conditions de construction et la préférence d'un style. Alors des solutions sont produites à l'aide d'un algorithme évolutionnaire. La démonstration d'une série d'expériences fournira l'évidence qu'un tel système est possible, faisable et efficace pour le début de la conception architecturale des bâtiments.

Cette étude se déploie autour des objectifs suivants:

Objectif 1 : Surmonter les problèmes de la représentation de la connaissance architecturale dans le cycle conceptuel déclaratif.

La représentation structurée et intégrée de la connaissance architecturale est importante parmi les différentes phases du processus de conception. L'intégration des systèmes de conception '*basés sur la connaissance*' et '*assistés par ordinateur*' fournit des outils nouveaux pour la synthèse de conception et stimule la créativité des concepteurs. Les concepteurs utilisent des ensembles de connaissance et certains outils opérationnels. La conception architecturale assistée par ordinateur dont l'aide est basée sur la connaissance vise à développer les environnements de conception à l'étape conceptuelle du processus de conception. Par conséquent, nous définissons un

cadre approprié afin de faire valoir la connaissance architecturale dans une base de connaissance de type modèleur déclaratif.

Objectif 2 : Surmonter les problèmes de la représentation du style architectural dans le cycle conceptuel déclaratif.

Le style est une sous - catégorie de la connaissance de conception. Le style aide le concepteur à prendre de nombreuses décisions importantes particulièrement au début de la phase de conception. L'impact du style est très important lors du processus de conception architecturale car il pourrait définir plusieurs directives essentielles pour la conception d'un bâtiment. L'introduction du style dans un processus de conception générative assisté par ordinateur pourrait être bénéfique pendant la phase conceptuelle de la conception du bâtiment. Un schéma représentationnel pour le style architectural s'attaque au problème de la quantification des caractéristiques de qualité. Il est possible d'impliquer une proposition qui surmonte de tels problèmes dans le cycle de conception déclaratif dans les phases de description et de génération de solutions de conception.

Objectif 3 : Synthèse du paradigme de conception évolutionnaire dans le cycle conceptuel déclaratif.

Pendant la phase conceptuelle de la conception architecturale, les problèmes sont caractérisés par leur imprécision et leur complexité. Les conditions de construction sont mal définies et contradictoires. Le concepteur devrait explorer l'espace de solutions pour obtenir des solutions alternatives de construction en épurant les conditions et les contraintes. La recherche des solutions de conception alternatives ne requiert pas de recherche exhaustive. Le concepteur préfère explorer (avec richesse et diversité suffisante) des « voies » de l'espace de solutions. En général, des algorithmes évolutionnaires tendent dans cette direction. Une étude du mélange des algorithmes évolutionnaires dans la modélisation déclarative permettrait le développement d'un algorithme évolutionnaire pour la génération et l'évaluation dans le cycle la conception déclaratif.

1.3. Approche et méthodologie de recherche

Le travail de recherches pour le développement du système de conception déclaratif évolutionnaire est passé par les étapes suivantes :

- Étude des potentialités de la modélisation déclarative. Adoption et adaptation d'un cadre de modélisation déclarative approprié.
- Étude des modèles de données de bâtiment afin de synthétiser un modèle approprié pour la représentation de la connaissance architecturale.
- Étude de style pour le développement d'un modèle de représentation pour le style architectural de bâtiments, qui permettent l'utilisation du/des style(s) dans la conception déclarative en tant qu'outil d'aide à la conception architecturale conceptuelle.
- Étude des algorithmes évolutionnaires afin d'employer un algorithme génétique multi-objectif approprié. Une telle technique sera mise en application dans un système prototype de conception déclaratif évolutionnaire.
- Mise en œuvre d'un système prototype MultiCAD spécifique comprenant la connaissance architecturale, le style architectural, l'algorithme génétique multi-objectif et l'interface graphique utilisateur (GUI).
- Définition d'un cadre expérimental qui utilise des cas de conception spécifique. Deux styles architecturaux sont choisis pour l'évaluation des parties du système développé.
- Essai des possibilités du système prototype afin d'évaluer son degré d'efficacité et de praticabilité.

1.4. Signification - Avantages Potentiels

Cette thèse présente une recherche, qui a obtenu des résultats satisfaisants basés sur des approches, des modèles, des représentations, des réalisations et des cas de conception s'attaquant à un problème de recherche essentiel et à des cas de conception architecturale.

La conception architecturale implique un éventail de problèmes et de solutions. Dans cette thèse, nous nous concentrons sur le problème de synthèse de bâtiments, limité à la composition spatiale et à la génération de morphologie. Les caractéristiques du problème sont les différents types de contraintes et la solution est la synthèse assignée. Les contraintes du problème sont un facteur essentiel dans la conception et définissent l'information décisive dans la génération de conception [Akin 96], [Simon 96]. De telles conditions auraient pu être fonctionnelles, des préférences esthétiques,

des codes de construction, etc. L'impact des ordinateurs dans la conception de bâtiments est indiscutable. D'ailleurs, beaucoup de systèmes de Conception Assistée par Ordinateur (*CAO*) permettent la conception d'objets par des moyens procéduraux tandis qu'elles fournissent l'optimisation des produits en maximisant beaucoup d'aspects fonctionnels de conception. Toutefois, des tentatives ont visé à développer des outils de *CAO* pour l'assistance de conception d'artefacts prenant en considération les aspects esthétiques.

Le système peut aider à la conception architecturale des architectes dans un certain nombre de domaines.

Le système prototype obtenu pourrait aider à la synthèse de bâtiments prenant en considération les aspects esthétiques et les critères stylistiques particuliers. L'architecte-concepteur pourra concevoir des constructions adaptées à un style architectural particulier.

Le système fournirait un cadre de description pour l'expression déclarative des contraintes de conception et avec peu de connaissance préalable.

Il fournit une méthode pour quantifier des critères qualitatifs tels que l'esthétique d'un bâtiment. Un modèle de style pourrait permettre la représentation de style. La quantification du style pourrait fournir aux architectes plus de critères de décision pendant la génération des solutions. D'autre part, une telle représentation pourrait permettre à l'architecte de modéliser ses préférences esthétiques et les faire évoluer. Elle pourrait augmenter les capacités de l'architecte-concepteur en produisant des concepts de bâtiments nouveaux.

La méthode adoptée pour la modélisation du style architectural a un impact important sur les études au sujet de la formation, la représentation et l'utilisation du style dans la conception architecturale. En général, l'application actuelle pourrait avoir une influence rétroactive sur la théorie de la conception.

1.5. Grandes lignes de la thèse

Cette thèse présente l'analyse, le développement et l'implémentation d'un système de conception génératif de modélisation déclarative, basé sur de conception évolutionnaire.

Le deuxième chapitre suivant l'introduction présente une revue de la littérature relative aux domaines de recherche fondamentaux de cette thèse. Ces domaines sont

au nombre de trois: modélisation déclarative, algorithmes évolutionnaires et connaissance architecturale. Par conséquent, le chapitre est divisé dans trois sections. La première section présente la modélisation déclarative, fournissant une étude des possibilités de modélisation déclarative. Y sont présentés les développements récents des modeleurs dédiés. Un système de modélisation déclarative approprié y est présenté, le système de MultiCAD. La deuxième section présente des concepts de conception évolutionnaire. Les capacités et les avantages des algorithmes évolutionnaire y sont présentés et les algorithmes génétiques y sont défendus. Les algorithmes génétiques (AG) sont expliqués selon leur base théorique tandis que nous nous concentrons sur des méthodes d'optimisation multi-objectives. En outre, un certain nombre d'applications d'AG dans la phase de conception sont présentées. La troisième section propose un aperçu des concepts de la connaissance architecturale et plus particulièrement, des problèmes et des contraintes en phase de conception architecturale. En outre, l'utilisation de la modélisation de construction ainsi que les différentes approches dans le domaine architectural y sont discutées. On y présente plusieurs applications et résultats actuels dans ce domaine. Une justification explicative d'un certain nombre de choix y est clarifiée.

Le troisième chapitre aborde le concept du style. Cependant, le chapitre se concentre sur le style architectural et fournit une définition opérationnelle.

Nous considérons un style architectural basé sur la connaissance de l'organisation spatiale et de principes morphologiques. Dans l'analyse, le style architectural est exprimé à l'aide d'objets et de contraintes, qui forment à leur tour des principes stylistiques. Il y est proposé deux catégories d'objets: spatiaux et structurels. Les conditions stylistiques prennent en compte le placement des objets et les rapports entre les objets. Le style architectural est composé à partir d'une catégorie qui détermine des contraintes topologiques et une catégorie qui détermine des contraintes formatives. La représentation du style est sémantiquement riche parce que l'information sur le style de construction est entremêlée avec l'information sur chacun des différents éléments de construction. En conclusion, deux exemples de styles régionaux de bâtiments de l'architecture vernaculaire grecque y sont présentés : celui de Santorini et celui de Metsovo.

Le quatrième chapitre présente nos propositions et est divisé en trois sections. La première section définit le cadre afin d'exploiter la connaissance architecturale dans la

base de connaissance de MultiCAD. Les concepts de trois axes de recherche de base y sont combinés: la modélisation déclarative de la scène, phase de conception architecturale et la gestion de la connaissance. Cet effort facilite le processus de développement de modèles de construction cohérents avec l'exploitation de la connaissance architecturale, placée dans le système dans une structure appropriée. Un cadre de *Connaissance Déclarative pour Modeleur de Bâtiments orienté vers l'Architecture (CDMBA)* y est développé. Le cadre identifie des rapports sémantiques requis pour définir les modèles de bâtiments orientés vers l'architecture. Le concept du *Modèle de Bâtiment Déclaratif Normalisé de la Scène (MBDN)* y est présenté. Les avantages sont doubles. Premièrement, l'espace de solutions est limité aux solutions valides par l'intégration de la connaissance dans un système déclaratif. Deuxièmement, il est adapté à la représentation de la connaissance de style architectural.

Dans la deuxième section, nous modelons et employons le style architectural dans le cycle conceptuel déclaratif de system de conception MultiCAD. Nous fournissons une expression de la connaissance de style architectural sous forme de critères stylistiques. Les critères stylistiques seront employés dans la phase de description et de génération du cycle conceptuel déclaratif. En outre, un nouveau modèle sémantique et géométrique sera incorporé afin de faire face aux demandes spéciales de la représentation de style. La structure développée pour représenter le style architectural répond à deux critères: L'information stylistique sera représentée indépendamment de la catégorie de style et la représentation de style sera conforme à un cadre, qui peut être facilement adapté aux différents styles architecturaux et incorpore toutes les connaissances différentes de style. Le cadre qui en résulte offre un instrument de base pour représenter les approches stylistiques architecturales et deuxièmement, il facilite la génération des conceptions alternatives (scènes) pendant la phase de conception en réorientant le processus de recherche de conception vers des demandes spécifiques.

La troisième section propose un algorithme génétique multi-objectif qui sera employé comme nouveau moteur de génération dans le cycle conceptuel déclaratif MultiCAD. Etant donné que les critères stylistiques seront considérés comme des critères objectifs dans un processus évolutionnaire, l'*Algorithme Génétique Multi-Objectifs (AGMO)* est un choix approprié. La manière par laquelle les principes stylistiques prennent la

forme de fonctions objectives est définie. Par conséquent, les principes stylistiques sont exprimés comme une équation linéaire multi-objective pour l'évaluation de style. Dans le cinquième chapitre, les deux initiales réalisations de propositions du début et le système courant de prototype sont présentés. Le chapitre est divisé en trois sections. Le contenu de la première section concerne l'application de la première tentative d'un algorithme génétique simple dans MultiCAD. Dans la deuxième section, présente le cadre pour la connaissance architecturale y est appliqué et les principes stylistiques sous la forme d'une bibliothèque de style y sont promus. La troisième section aborde l'application principale de la thèse.

Le développement d'un système évolutif dans MultiCAD y est défini. Le système pourrait faire évoluer les conceptions architecturales en termes de composition spatiale et d'expression morphologique sous des critères objectifs stylistiques. Les conceptions de bâtiments sont adaptées jusqu'à un certain degré à un style architectural.

Le système fournit les deux procédures distinctes qui se servent du système de conception déclarative évolutionnaire, le *développeur* de style et la procédure de *concepteur*. La première concerne un environnement pour l'expression de principes stylistiques en tant qu'équation linéaire avec des facteurs de poids. Nous permettons au *développeur* de fournir des poids pour chaque critère jusqu'à ce qu'il/elle soit capable de décider quels critères sont les plus appropriés pour l'expression d'un style particulier. Ces poids représentent l'importance relative de chaque principe stylistique. Le *développeur* place également les valeurs appropriées pour les paramètres de l'algorithme génétique. Du point de vue de *concepteur*, il/elle utilise uniquement des styles architecturaux spécifiques. Le concepteur pourrait seulement changer les paramètres de l'algorithme génétique pour obtenir un meilleur rendement. De cette façon, le système s'est étendu à la sphère de la recherche de l'adaptation de la morphologie de construction au (x) style (s) architectural (aux) spécifique (s). Un bâtiment développe son aspect esthétique, en termes de définition de sa forme globale, tout en composant ses espaces sous des critères stylistiques afin de faire évoluer la forme du bâtiment dans un style architectural. En parallèle, des critères de styles différents pourraient être combinés pour l'apparition de style (s) architectural (aux) nouveau (x). Le prototype est équipé d'interface graphique utilisateur (GUI) facile à utiliser.

Dans le sixième chapitre, une série analytique d'expériences est proposée pour l'évaluation du système prototype. En outre, un cadre expérimental utilisant des cas de conception spécifiques est défini. Ce cadre inclut deux cas de style architectural, qui sont choisis comme appropriés pour l'évaluation des parties de système de prototype. Les styles sont ceux de Santorini et de Metsovo. Ils sont choisis parce qu'ils ont des compositions spatiales différentes et des formes de toits différents. La méthodologie comporte deux étapes. Dans la première étape, la composition spatiale d'un bâtiment est développée. Dans la deuxième étape, la morphologie du toit évolue pour une synthèse de bâtiment choisie dans l'étape précédente. La procédure est répétée pour les deux styles. Les métriques spécifiques sont obtenues pour contrôler le rendement des paramètres de l'algorithme génétique multi-objectifs. De plus, le système surveille également le rendement de la somme d'objectifs pondérée. Cette activité est importante pour l'expression exacte des critères de chaque style dans la méthode d'agrégation. Les exemples des styles architecturaux ont été introduits pour une description de bâtiments donnée. De cette façon, le rendement du système est évalué. Les exemples de conception de bâtiments qui en découlent illustrent l'efficacité et la faisabilité du système de prototype pour la conception stylistique conceptuelle de bâtiments.

Dans le septième chapitre, une discussion et des conclusions y sont présentées. Il deviendra évident que quoique de nombreuses conclusions puissent être tirées du travail et des expériences développées, beaucoup de nouvelles questions sont apparues pendant le processus. Un certain nombre de questions de recherche émergeaient en vue de futures directions de recherche.

Chapter 2

General Review of Literature

2.1. Introduction

In this chapter we will describe the main research areas of this dissertation. This research work defines an emerging research area, which integrates Evolutionary computing techniques with Declarative Modelling in the Computer Aided Architectural Design (*CAAD*) area. This chapter reviews and assesses the related significant research in these areas. Given the reason that our main aim is the representation of architectural style for the aid of the conceptual phase of architectural design we will argue for two directions. The first direction is the requirement to introduce domain-specific knowledge, that of architectural design, in a declarative modeller. Secondly, in order to confront certain problems and drawbacks of the current generation engine we support the need to incorporate a new generation engine for the generation phase of a declarative modeller.

In the first section we will provide a detailed presentation of the declarative modelling paradigm. The general structure of declarative modellers will be presented. We concentrate our study on how some limitations were confronted in recent applications of declarative modellers. Next, we present a declarative modelling framework appropriate for our research methodology. That framework will be the MultiCAD-II and will be adapted to our requirements.

In the second section we endow with a presentation of the theory of evolutionary algorithms, and in particular the paradigm of *Genetic Algorithms (GAs)*. Especially we focus our interest on multi-objective optimisation GAs. We will underline the impact of GAs systems during conceptual design while we present a number of recent applications of successful implementation of GAs in conceptual design.

In the third section we present analytically the area of architectural design knowledge modelling in CAAD systems. In particular, we will present recent advances from the field of Building Product Modelling and Feature-based Modelling.

2.2. Declarative modelling

In general, a modeller can be viewed as a set of tools dedicated for the manipulation of geometric objects. These tools could be classified in two main categories:

- Interactive tools, whose inputs are the mouse, a graphic tablet, a glove, et cetera,
- Parametric tools, whose inputs are mainly numerical.

In most of the interactive tools the designer needs to be able to execute precise movements. Parametric tools require from the designer to have a sufficient knowledge in advance of the underlying geometric model in order to input the right values. It is evident that in both cases, the designer is expected to have a very high exactitude. The designer must have prior knowledge on exactly all that he/she wants to create and how to do it. That knowledge up to a certain degree is strongly inevitable and obligatory. These modellers appear to be powerful and useful for a set of problems, however their tools may not fit all the designer's expectations. The generation and exploration of *scenes*, i.e. geometric objects, based on incomplete design requirements, and the handling of a multitude of variables and constraints embedded within scenes are challenging tasks for designers. Especially at the beginning of the creation of a scene the designer has no flexibility in the description of scene requirements. Moreover, the designers confront the task of verification of scene parameters. Finally, the designer does not have the opportunity to obtain alternative solutions for his/her problem.

An approach that confronts successfully and attenuates drawbacks and limitations of classical geometric modelling is declarative modelling. The declarative modelling paradigm introduces property based modelling techniques by providing the possibility of scene description using properties, which can be either precise or imprecise [Plemenos 91], [Plemenos 95], [Lucas 90]. It is an intuitive modelling tool that does not require the same exactitude as classic modelling tools. Declarative modelling is a total approach of the designing process [Plemenos 02]. Declarative scene modelling allows the description of scenes by only giving some expected properties of the scene and letting the modeller find alternative solutions, if any, verifying these properties. The declarative modelling process is made of three phases: the description phase, where the designer describes the scene, the scene generation phase, where the modeller generates one or more scenes verifying the description, and the scene

understanding phase, where the designer, or the modeller, tries to understand a generated scene in order to decide whether the proposed solution is a satisfactory one, or not.

When a designer describes a scene in an intuitive manner, with the use of common expressions, the described properties are in many cases imprecise. In general there appear two kinds of imprecision. The first type appears when numerous values can satisfy a property. For example, a user can tell a modeller that ‘*the scene A must be placed on the left of scene B*’. There exist several possibilities to place a scene on the left of another one. The second kind of imprecision is due to the fuzziness of a property [Plemenos 02]. The designer lacks the exact property that his/her scene has to satisfy and, therefore, expects some proposals from the modeller. Hence, the designer can indicate that ‘*the room A must be near the room B*’ without giving any other precision. In order to enable this lack of precision, declarative modelling pays a high price: it is generally a time consuming scene modelling technique.

2.2.1. Advantages of declarative modelling

The declarative modelling provides the designer with a more familiar way and has significant advantages. The main disadvantages of declarative modelling are caused by its advantages. In the following we present some principal advantages and disadvantages of declarative modelling.

Intuition. Intuition is an essential quality of a declarative modeller. Declarative modelling makes it possible to approach the design of a scene in a more intuitive way. A declarative modeller provides concepts of higher degree for scene description. Concepts are very near to a natural language. These concepts belong to the domain of a dedicated declarative modeller.

Automatic inspection of scene properties. The user provides the declarative modeller with the description of a scene. The description has all properties and relations between the objects of a scene. Then the modeller handles all information and it tests and verifies all the criteria integrated in a description. Next, a scene generator module automatically generates scenes that satisfy all given properties and relations.

Multi-Solutions. Declarative modelling is a top-down technique that does not offer only a single solution but a model from which the system computes one or more solutions.

Handling non-geometrical aspects. Declarative modelling enables the description of a scene on a higher level. Declarative modeller allows the use of precise and imprecise descriptions, and it is not based on the complete knowledge of the underlying geometric models and/or the designer's skill. Therefore a declarative modeller can treat other aspects of the scene apart from purely geometrical or graphic aspects. In advance is independent from the level precision that it is used.

Adaptation to design process. Declarative modelling is well adapted to the design process. It provides the approach of an object from various points of interest. It can unify in an excellent way the development of scenes that meet multiple desired properties.

2.2.2. Limitations of declarative modelling

Description Interpretation. Declarative modelling poses a significant problem, the interpretation of description. A multiplicity of possible interpretations sometimes contradictory or incoherent was often induced with handling of high level concepts. The concept of interpretation is often subjective, and the majority of the declarative modellers approach that problem only partially. Nevertheless the effectiveness of a declarative modeller depends on the adequacy between the interpretation of the user and that of the modeller.

Generation-understanding phase. In order to search for the scenes, by verifying the properties of a description, declarative modellers use the exploration of a finite but often very large universe of possible forms. This research can lead the user towards solutions which he/she did not consider. However, there can be a very significant number of solutions according to the precision of description. A great number of solutions are produced, a lot of them very similar, their number being well beyond what the user can manage to properly evaluate them.

2.2.3. Declarative modellers

There is a great variety of declarative modellers and all are oriented on specific tasks. Declarative modellers are characterised by the kind of the smallest element that can be characterised in the description given at the input of the modeller, and the kind of relations between these elements, and those provided at the output of the modeller.

A declarative modeller is the implication of the three phases of the declarative modelling [Lucas 95]:

1. *Description phase*. During that phase, the user is offered an application language and an interface to describe the properties of scenes or objects. High-level descriptions are translated in an internal language in order to become easier to manipulate and process. In particular, the designer describes the topology, geometry, et cetera of the objects of a scene in a declarative way by means of properties and constraints on objects and their configuration.
2. *Generation phase*. It is the second phase and it can be considered as the system's essential phase. During this phase vast search spaces are explored for the production of models that satisfy the description made by the designer even if it is incomplete. One of the key ideas in declarative modelling is the ability of the system to find several or all solutions. Then, given a consistent description, a generation engine produces at least one of the numerous solution scenes related to this description and potentially all the solutions.
3. *Understanding phase*. Many solutions can be generated according to the designer's specifications. This phase permits visualisation, navigation and modification of valid models to facilitate the selection of a solution by the user.

A great number of declarative modellers have been developed mainly in France [Martin 88], [Poulet 96], [Colin 97], [Kwaiter 98]. However, most of them are limited to very restricted domains. A domain specific declarative modeller can be very efficient because its scene generator has been developed to be well adapted to that domain. We present some examples of declarative modellers.

DES²MON is a modeller whose objective is to assist the design of three-dimensional scenes with regard to dimensioning and placement of objects in a finite universe. The modeller provides to the designer a library of objects (describing their principal characteristics) and space constraints (to manage the relative positioning of the objects), stated in a high-level language. This language allows the expression of the description of the scenes in natural language, for example '*the chair is beside the table*' [Kwaiter 98].

BatiMan is a modeller that deals with the construction of buildings decomposable to a finite set of elements by introducing methods of training [Champiaux 98a]. Is a modeller related to the domain of architecture and in particular, space organisation. The designer describes in restricted natural language, the components of the product in design and their positioning, proportions and relative aspect. The generation engine is also based on the technique of Constraints Satisfaction for the production of the

representations of buildings as volumes. *BatiMan* uses methods of incremental training in order to reduce the space of the solutions generated. This happens because it utilises the acquired knowledge on similar problems. Certain semantic observations or geometrical characteristics, on declarative descriptions, consists the criterion of classification of knowledge. In this way, from a representative scene, the modeller provides the valid solutions via a user interface of exploration of the solutions. Champciaux has confronted a major problem of declarative modelling [Champciaux 98b]. During the understanding phase of declarative modelling the large amount of resulted solution scenes provides many difficulties to the user of the system. In order to overcome this tedious and hard task he proposed a classification tool that automatically categorise solutions. In this way solutions resulted in classes hierarchically structured. Each class clusters similar solution scenes. Finally two tools were provided to the designer. One for browsing in a class hierarchy and another that provides an overview of all solution classes.

2.2.4. General and dedicated declarative modellers

Declarative modellers could be classified in to categories, generic and dedicated modellers. The declarative modellers are defined for a well delimited modelling area. Generic modellers are mainly domain-independent of any knowledge domain. Dedicated modelling is focused on a specific application domain. Therefore the principle of dedicated modelling is to define a declarative modeller for a well delimited modelling area.

The advantage of dedicated declarative modellers is efficiency because their solution generation engine can be well adapted to the properties of the specific modelling area covered by the modeller. On the other hand, it is difficult for such a modeller to evolve in order to be able to process another specific modelling area.

The aim of the general purpose modellers is generality. These modellers include a solution generation engine which can process several kinds of properties, together with a reduced set of pre-defined properties, as general as possible. General purpose declarative modellers could normally be specialised in a specific modelling area by adding new properties to them, corresponding to the specific modelling area we want to cover. In this sense, general purpose modellers can be seen as platforms to generate dedicated declarative modellers. The main advantage of general purpose declarative modellers is generality which allows specialising a modeller in a specific modelling

area without having to modify its solution generation engine. On the other hand, general purpose modellers suffer from their lack of efficiency, because of the generality of the solution generation mechanism [Plemenos 02].

2.2.5. Declarative Modelling by Hierarchical Decomposition

Declarative Modelling by Hierarchical Decomposition (*DMHD*) is a special approach of declarative modelling. It is used in order to model complex scenes [Plemenos 91], [Plemenos 93], [Plemenos 98], [Bonnetfoi 02]. Whereas the declarative modelling paradigm only defines the working mode of declarative modellers, the *DMHD* defines a top-down approach for declarative scene modelling at different detail levels. The structure of a scene can easily be represented using a hierarchical decomposition tree. Such a representation is commonly used in many modellers. The *DMHD* also uses a decomposition tree for the description and the generation of a scene. As a result the description can be an iterative process following a top-down approach. This allows a scene description and generation at various detail levels that can be made in one step or more. The major advantages of the declarative modelling by hierarchical decomposition are the following:

- Scenes are expressed in a gradual manner by the designer.
- A scene could be specified at different levels of detail.
- Allows interaction with the users for guiding the generation process: the user can choose a partial solution according to a particular depth level in the tree (similar to folding some nodes into leafs), and from this solution as mould, he/she can obtain complete solution. This is a special kind of interactivity based on the levels of detail.
- Enforces the locality of the description (the designer's scope is focused on explaining a part of the whole scene without thinking about other parts).
- Allows factorising of the properties.
- Induces strong relationships between a scene and its sub-scenes (inheritance of constraints for a node from its parent: the bounding box of each sub-scene is included in the bounding box of the scene from which it arises).
- Defines a strong independence between elements of sub-trees issued from the same node of the hierarchical decomposition tree: properties can only be expressed on features of only one node, or one node and its direct children nodes.

2.2.5.1. DMHD modellers

The declarative modelling by hierarchical decomposition has been introduced in order to propose general-purpose declarative modellers [Plemenos 91]. MultiFormes is a prototype of a declarative modeller using *DMHD* in development at the laboratory of *Méthode et Structures Informatiques* (MSI) of the University of Limoges.

2.2.5.1.1. MultiFormes

MultiFormes models complex scenes (for example dwellings) described according to the technique of the hierarchical decomposition [Plemenos 95], [Bonnefoi 99], [Ruchaud 01]. According to this technique, in order to describe a scene, it is broken up into parts which are recursively described (up to a certain level of detail).

The declarative modelling by hierarchical decomposition is specific to MultiFormes [Plemenos 95]. This technique, employs top-down description and criteria of complexity to partition the scene in various sub-scenes: When a scene is too difficult to describe with the language recognised by the modeller (it requires a too large amount of properties), it is partitioned recursively into two or more sub-scenes easier to describe, in accordance with the logical and spatial structure of the whole scene. An example is given below, where a '*hierarchical decomposition tree*' is associated to the description: a node of the hierarchical decomposition tree (the node '*Chair*' for the scene) is related to each scene of the description. For each node, the set of all sub-scenes produced by its decomposition corresponds to the set of children nodes. In the case of MultiFormes the elements handled at the input and output are bounding boxes. So, a scene produced by MultiFormes is constituted of '*objects*' given basically by their bounding boxes. Each object is handled by its isothetic bounding box. MultiFormes can also be used to generate any kind of scenes made of isothetic boxes. As the only supported geometric primitive is the isothetic box, MultiFormes uses a simple parametric representation of it. Each of these bounding boxes, is described by a position (x , y and z coordinates) and a displacement vector defining the width, the height and the depth of the associated bounding box (eventually some other relevant information are associated like the colours or the texture of the object). Each bounding box is described by at least six numeric variables. So, properties in MultiFormes are essentially related to position and size of one bounding box in relation with others. These properties are applied on the variables characterising bounding boxes. In order to limit the possible values of all variables, a workspace is

defined. All boxes are constrained to take place into the workspace. Until now, the scene generator module of MultiFormes has used a numeric constraint solver. Several research works has been done concerning this solver [Tamine 95], [Plemenos 98], [Bonnefoi 99].

The work of the declarative modeller consists in generating all possible scenes verifying the given high-level properties. In the case of MultiFormes, the generation of a solution that satisfies all the properties is obtained through the search of values for all the variables appearing in the scene (at least six variables for each bounding box). In order to allow computation of all possibilities, the different variables characterising the scene must take a finite number of values (a bounded set of integers for example). The exhaustive exploration of all possible values towards the achievement of a solution, combined with a large amount of bounding boxes (therefore a large amount of variables) and an imprecise description results in a prohibited amount of time required in order to obtain a solution scene.

The work of two researchers has confronted with limitations and drawbacks of the description and generation phase of MultiFormes [Bonnefoi 99], [Ruchaud 01].

2.2.5.1.2. Declarative Modelling of Habitation Edifices

Fribault has as a theme the declarative modelling of habitation edifices. Fribault proposed an information system associated with a declarative modeller for the assistance of architectural design. She presented the structure of the tables of the data base dedicated to the information system proposed and the possibilities of integration of this whole in a MultiCAD type system. Furthermore, it is determined the informational framework of a particular project in the field of building design. Next, the necessary properties were defined for the description of the basic component of a building (liveable space) with the various types of assembly rules utilised in the design according to the context. She exposed the preliminary stage of diagnosis which constitutes a first checking of coherence of initial description. Finally, she proposed a phase of generation based on the GNU-Prolog compiler which deals with the resolution of constraints on finite domains, [Fribault 98], [Fribault 04].

2.2.6. MultiCAD II

MultiCAD-II is a software architecture framework for the development of multimedia and intelligent information systems in order to support declarative design processes

[Miaoulis 02]. The current solutions engine of prototype MultiCAD-II is that of the MultiFormes project based on constraint-programming techniques. MultiCAD-II uses constraint-programming techniques to generate solutions, i.e. 3D forms satisfying the user's constraints. The main disadvantages of such a search strategy is: a) its exhaustive sequential search nature leading to unacceptably long times in relatively large problem spaces; and b) its inability to interact with the user and derive solutions that satisfy his/her aesthetics.

2.2.6.1. MULTICAD-II Framework

MultiCAD-II was the first attempt to create a complete conceptual modelling environment that would help users' implementation of three-dimensional models through an abstract modelling language. Users do not have to deal with dimensional details. The computer generates all possible solutions of a given model and the users' task is to evaluate each solution according to their personal criteria. MultiCAD-II is able to read *Internal Model Description* (IMD) script files and analyses them in order to create graphic solutions. The solutions are displayed one by one on a graphics screen, and the user has to either accept it or move on to the next solution. In MultiCAD-II, the solutions are calculated using constraint-programming techniques. Constraint programming is a full search algorithm, which scans the entire search tree exhaustively. It uses a repetitive generate-and-test method to access each valid tree node sequentially. Each node is checked whether it meets the model's restrictions. This algorithm is easy to implement using a programming language. The mathematical foundation is relatively simple, because it uses plain comparison operators for calculating bounding boxes for the scenes. Due to the fact that constraint programming is a sequential method, such a technique is slow and inefficient for large search trees. In addition, the user has to evaluate each solution before he/she moves on to the next one.

2.2.6.1.1. MultiCAD-II Software architecture

The directions of MultiCAD-II software architecture framework are defined through a research project supported by the Laboratory MSI of the University of Limoges along with the Intelligent Information Systems Engineering Research Team of Informatics Department of TEI (Technological Education Institute) of Athens [Miaoulis 96].

MultiCAD-II is a multi-layered architecture that comprises the following main layers [Miaoulis 02]:

- The *Interface layer* encompasses functions such as intelligent visualisation of scene models and documents, creation and editing of models and description, formulation of the request (traditional formulations of SQL, spatial SQL or free text search), navigation and browsing of databases, acquisition and editing the different types of knowledge and information, application and interaction control.
- The *Process layer* comprises functions such as generation or understanding between the different levels of models, converting the different types of the same level's models.
- The *Information and Knowledge Management layer* is used for structuring, management, searching and exploitation of the different databases.

These layers are projected to the phases of the declarative conception cycle (Figure 2-1).

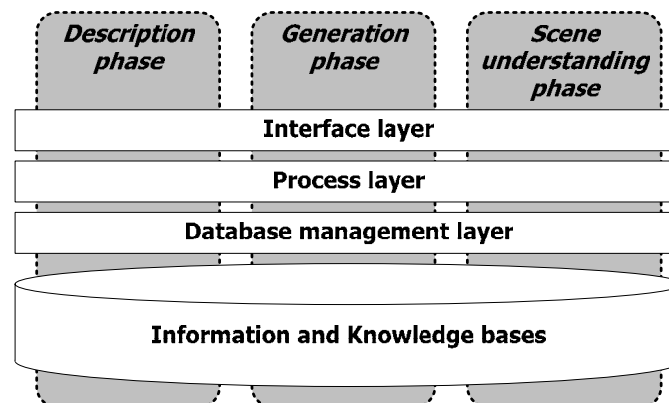


Figure 2-1 MultiCAD-II layered architecture

(Following the declarative conception cycle)

Description phase

The designer describes in a high-level of abstraction his desired scene by defining the scene's decomposing *objects* (entities), their *properties* and *relations*. The scene's description is represented in the following way:

- *Tree-formed*. This form explicitly represents the hierarchical decomposition of the scene,

- *Text-formed representation*. It is expressed in a formal Prolog-like language [Plemenos 95], representing the Internal Declarative Representation – IDR of the scene.
- *Representation in a relational-object database* [Miaoulis 02]. The scene's description is stored as an assembly of objects having properties, and being related to each other through relations.

The storage system is organised in four logical databases:

- The *Scene database* supports information describing the scene models (IDR models, relations between the composing objects et cetera).
- The *Project database* manipulates data concerning the planning, the finance and other special oriented data for each project.
- The *Multimedia library* containing all types of documents related to a project's files, (geometrical models and multimedia information as pictures and videos).
- The *Knowledge base*, where exist all necessary information about entities' types, their properties and relations.

The *Scene database* is configured following the scene *Conceptual Modelling Framework* (CMF), proposed in [Miaoulis 00], where the description of a scene may contain:

- Simple objects (defined by their concrete properties), or
- Generic ones (group of simple objects having properties in common).

There exist three types of relations between the objects:

- Meronymic (*part-of, is included in*),
- Spatial organisation (*near by, left to*), and
- Correlation (comparison of objects *higher than*).

The latest version of the conceptual model of MultiCAD-II architecture [Miaoulis 02] is based on the *Extended Entity-Relationship* (EE-R) model: It extends the *Entity-Relationship* model theory in order to include notions as aggregation, inheritance et cetera. The scene's description is stored in the relational-object database as an assembly of objects having properties and being related to each other through relations. Furthermore, the *Knowledge Base* contains a series of types of objects,

types of properties and types of relations constituting the description of Knowledge in MultiCAD-II. The elements of the Scene meta-model are instantiated in the Knowledge Base.

Generation phase

Many attempts have already provided the user of MultiCAD-II with alternative solution generators. Each generator results in alternative solutions of models that satisfy the constraints specified from the scene's description. In particular, there exist mechanisms, which aid the generation of alternative solutions with the use of Constraint Satisfaction Problem solvers [Bonnetfoi 00], Genetic Algorithms [Vassilas 02], and Neural Networks [Plemenos 02]. In this presentation we will only present the Constraint Satisfaction Problem approach. The two remaining methods, Genetic Algorithms and Artificial Neural Network (ANN) will be presented in the foreground on the thesis research.

Scene understanding phase

The generated scene solutions are visualised through their geometrical representation: Bitmaps [Miaoulis 02], VRML [Vassilas 02], and AutoCAD designs [Makris 03]. The designer can evaluate the solutions according to his/her criteria either by selecting the best ones or by setting a score to each of them [Vassilas 02].

2.2.7. Discussion

The study of particular recent approaches and their applications reveals specific limitations and drawbacks in the description and generation-evaluation phase of dedicated declarative modellers. We focus our research attention on:

- The difficulty in the description of complex scenes and
- The enormous number of resulted solutions during the generation phase.

Specific proposals were applied and evaluated by many researchers. The evaluation of the development systems is very promising. Although there was an obvious improvement in the parts of declarative modelling these two problems remain as open research issues.

2.2.7.1. Description phase

In a design case-problem the use of imprecise properties would increase the richness of the solution space and would allow the user to obtain concrete answers for a vague mental image. So, the use of imprecise properties plays an important role for the designer. On the other hand, imprecision should not exceed a particular threshold as defined by the domain-specific knowledge. Otherwise the declarative modeller will not efficiently aid the designer.

2.2.7.2. Generation-understanding phase

A declarative modeller provides two different working modes: *exploration* and *solution search* mode. When a declarative modeller is used in exploration mode, it starts from a designer's description, performs a full exploration of the solution space and offers the designer all found solutions. This mode is adequate in two cases. First, when the designer has insufficient knowledge of a domain and wants to discover it by an exhaustive exploration. Second when the designer wants novel ideas and hopes that the modeller could help him/her by exploring a vague description. During the exploration mode it is crucial to apply techniques for exploration cost reduction by reducing the number of ineffective tries during the solution search process [Plemenos, 97], [Plemenos 98], [Bonnefoi00]. The exploration mode has an inherited problem because of the use of general imprecise properties. Enormous numbers of solutions are produced and this causes difficulties in their management. Furthermore, some categories of solutions are of no interest for the designer. So it would be very supportive if the designer could exclude the generation of such solutions in subsequent generations. In solution search mode, the designer has a relatively accurate initiative of the kind of the resulted scenes. In many design cases the designer would like to obtain a solution immediately or very quickly from a description using less imprecise properties. Because of the semantic ambiguity of a property the modeller can not overcome production of unfeasible solutions.

Up to now suggested approaches depend on knowledge of the designer's preferences which can guide the modeller in its search. The modeller learns to avoid the examination of solutions that would not satisfy the designer intuition. These approaches use techniques from machine learning for the improvement of declarative modelling in both exploration and solution search mode. As the modeller does not

know the designer's preferences, machine learning can be used to teach the modeller what kind of scenes are, or are not, motivating [Plemenos 93], [Plemenos 98].

An interactive machine learning mechanism has been implemented for the improvement of exploration and solution search mode. The mechanism is based on neural networks, and it was implied in a DMHD modeller. The modeller uses the mechanism in two steps. Firstly, during a learning phase, some scenes, generated by the modeller from the initial description, are selected by the user to serve as examples of wished scenes. Each time a new example is presented, the modeller learns more on the user's preferences and this is materialised by a modification of the values of weights associated to the connections of the network. At the end of the learning phase, an acceptance interval is calculated and assigned to each decision cell. Secondly, after the end of the learning phase, the modeller is in the normal working phase: the weights of connections calculated during the learning phase are used as filters allowing the choice of scenes which will be presented to the user. The used machine learning mechanism takes into account only relative dimension and position properties of a scene. Form properties are processed by the scene generation engine after selection of dimensions and positions of the bounding boxes of each sub-scene. The neural network gives satisfactory results. It acts with little information and already learnt knowledge can be used for continuous machine learning. Furthermore the modeller avoids more and more non interesting scenes. The achieving of this interactive machine learning is the reduction of the number of solutions shown to the user. Unfortunately the search space has not diminished. Therefore there is no decrease in the number of tries, and the total exploration time has not changed [Plemenos 02].

In recent applications there have been introduced an intelligent adaptation to user preferences within the MultiCAD-II context [Bardis 04], [Bardis 05a], [Bardis 05b]. This approach employs a combination of a Decision Support component as well as a Machine Learning (ML) component in order to acquire and maintain user's preferences in the form of a user profile. This profile is applied to the set of generated solutions in order to select those that are closer to the user's preferences. The performance of the two components is compared with the actual user's selections. The latter provide further training to the ML component until it outperforms the Decision Support component and eventually becomes exclusively responsible for solution selection.

An alternative approach in order to confront the abovementioned problems of generation-evaluation in declarative modelling is Genetic Algorithms (GAs). GAs based machine learning has been implemented on MultiCAD-II an information system for CAD, which is based on the generation engine of MultiFormes for scene generation [Vassilas 02]. In this implementation, only some solutions, the initial population, are generated using the time consuming constraint satisfaction techniques, used by the main generation engine of MultiFormes. The final solutions are obtained by the evolution process used in GAs, with the search being guided by the user on more promising parts of the search tree [Plemenos 02].

A similar approach has appeared making use of GA in another declarative modelling platform, DEM²ONS (Declarative Multimodal ModellIng System) for 3d object layout. It is a general-purposed constraint-based system. The constraint solver was based on a GA. With the use of basic and complex sets of constraints combined with Boolean trees the system provides promising results using examples of complex scenes [Sanchez 03].

A research approach towards the aid of the scene description concerns the introduction of the notion of '*concept*' in the declarative conceptual cycle [Ravani 00]. In this case the designer could have at his/her disposal prior knowledge structures. The research work [Ravani 04] concerns the representation of '*concepts*' in declarative design and the development of a subsystem that deals with concept definition, evolution and ontology re-arrangement. That work is based on the domain of Architecture. The subsystem aids the designer at the conceptual phase of declarative design in classifying design elements as domain concepts, to reuse already registered (in a structured manner) past knowledge of the domain (in form of past cases and concepts), by using available ontology manipulation tools. The subsystem supports concept evolution and ontology reorganisation processes. Furthermore, a domain expert can interfere and edit the concept hierarchy apart from the design description phase.

2.3. Evolutionary Search

The second section provides a presentation of the evolutionary techniques with a special focus on the Genetic Algorithms (GAs) while we argue about the selection of GAs.

There are many search algorithms known in computer science, of which evolutionary search is a small and recent sub-set. Search algorithms define a design problem in terms of search, where the search-space is a space filled with all possible solutions to the problem, and a point in that space defines a solution [Kanal 88]. The problem of improving a design is then transformed into the problem of searching for better solutions elsewhere in the space of allowable designs.

Evolutionary search algorithms are inspired by and based upon evolution in nature. These algorithms typically use an analogy with natural evolution to perform search by evolving solutions to problems. Hence, instead of working with one solution at a time in the search-space, these algorithms consider a large collection or population of solutions at once. Evolution-based algorithms have been found to be some of the most flexible, efficient and robust of all search algorithms known to Computer Science [Goldberg 89]. Because of these properties, these methods are now becoming widely used to solve a broad range of different problems. In the domain of design, the use of evolutionary search to optimise existing designs is becoming widespread [Holland 92].

During the last three decades there has been a growing interest in algorithms that rely on analogies to natural processes. The emergence of parallel computers with massive process power made these algorithms of practical interest. The best-known algorithms included:

- Evolutionary Programming,
- Genetic Algorithms,
- Evolutionary Strategies,
- Simulated Annealing, and
- Classifier Systems.

A term '*evolutionary program*' is used to define a subclass of algorithms, those that are based on the principle of evolution, survival of the fittest [Michalewicz 92]. In evolutionary algorithms a population of individuals, the potential solutions, undergoes a sequence of transformations using the mutation and crossover operators. These individuals strive for survival: a selection scheme, biased towards fitter individuals, selects the next generation. After some number of generations, the program converges guided by selection based on the predefined fitness function or diverges with random

selection. Of particular relevance to the present work is the development of evolutionary design that is based on the use of the adaptive search technique known as the Genetic Algorithm. GAs have been increasingly recognised in various branches of engineering, both as a powerful tool for the detailed design of complex components and as a high-level decision support technique [Bullock 95]. In general, GAs demonstrates the basic principle of these algorithms. And they are the best known and the most widely used evolutionary techniques [Holland 75], [Davis 91]. This process resembles the natural evolution process of Darwinism, of which the selection, transmission and variation are three main ingredients [Dawkins 83]. In GAs a population of solutions to the problem is maintained, with the ‘*fittest*’ solutions (those that solve the problem best) being favoured for ‘*reproduction*’ every generation, during an otherwise random selection process. ‘*Offspring*’ are then generated from these fit parents using random crossover and mutation operators, resulting in a new population of fitter solutions [Holland 75]. There are four characteristics that make GAs differ from traditional algorithms [Goldberg, 89]:

- GAs usually works with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use objective function information, not derivatives or other auxiliary knowledge
- GAs use probabilistic transition rules, not deterministic rules.

2.3.1. Genetic Algorithms

The popularity of GAs is based on their ability to tackle a huge variety of optimisation problems (including discontinuous functions), for their consistent ability to provide excellent results and for their robustness [Holland 75], [Goldberg 89], [Davis 91], [Fogel 95]. For a search algorithm to be robust, it must be capable of producing good solutions to a broad range of problems. Goldberg makes comparisons between traditional search methods (calculus-based, enumerative, and random) with GAs. He concludes: “*while our discussion has been no exhaustive examination of the myriad methods of traditional optimisation, we are left with a somewhat unsettling conclusion: conventional search methods are not robust*” [Goldberg 89]. Although this point of view is generally not in agreement from many researchers, and it is

significantly argued that a traditional algorithm designed specifically for a problem will provide better results for that problem than a GA could, but that a GA will provide good solutions for a much broader selection of problems, compared to such problem-specific methods [Fogel 95].

GAs have become widely used for a broad range of optimisation problems in the last ten years [Holland 92]. One of the pioneers in the research of GAs Goldberg suggests them as being a “*search algorithm with some of the innovative flair of human search*” [Goldberg 89]. Another of the advantages of GAs is that they are very forgiving algorithms –even if they are badly implemented, or poorly applied, they will often still produce acceptable results [Davis 91]. GAs achieves much of their expansion by ignoring information except that concerning payoff. While other methods rely heavily on such information and in problems where the necessary information is not available or difficult to obtain, such as some design problems, these other techniques break down. GAs has similarities in the underlying coding together with information ranking the structures according to their survival capability in the current environment. By exploiting such widely available information, GAs may be applied to virtually any problem. The transition rules of GAs are stochastic, while many other methods have deterministic transition rules. GAs use random choice to guide a highly exploitative search.

A GA is a search technique adequate for searching noisy solution spaces with local and global minima. Because it searches from a population of points, not a single point, the probability of the search getting trapped in a local minimum is limited. GAs start searching by randomly sampling within the solution space, and then use stochastic operators to direct a ‘*hill-climbing*’ process based on objective function values [Goldberg 89]. Despite their apparent simplicity, GAs have proved to have high efficacy in solving complex problems which other, more conventional optimisation methods, may have difficulties with, namely by being trapped in local minima.

The advantages of GAs are the following. They require no knowledge or gradient information about the objective function, and they can handle discontinuities. They can give information about the stability of a solution. GAs can be used to give solutions for complex multi-dimensional optimisation, for which few other techniques can give any information.

The GAs have disadvantages in the following cases. They do not necessarily find the exact global optimum. They require large number of evaluations of the objective function. In general, it is necessary to incorporate problem specific information for optimal performance. GAs requires a coding of the problem.

Experimental results show that for most GAs (initialised with random values), evolution makes extremely rapid progress at first, as the diverse elements in the initial population are combined and tested. Over time, the population begins to converge, with the separate individuals resembling each other more and more [Davis 91]. Effectively this results in the GA narrowing its search in the solution-space and reducing the size of any changes made by evolution until eventually the population converges to a single solution [Goldberg 89]. More advanced literature is given in [Holland 75]. A GA for solving a problem must have 5 mechanisms: chromosomal representation, population initialisation, evaluation function, recombination operators and values of the parameters. A short description of these mechanisms is given below. It mostly addresses Simple Genetic Algorithms (SGA).

2.3.1.1. Chromosomal representation

The genotypes have the form of different representation and encoding schemes- genotypes can be binary strings, real value strings or other possible forms [Goldberg, 89]. Binary encoding is preferred because of its simplicity and functionality. Furthermore, the use of bit strings, as well as being computationally convenient, simplifies the design and implementation of genetic operators such as crossover and mutation. The main advantage is that binary encoding is a good general, non-domain specific representation that is '*robust*' across a wide range of problem areas [Davis 91]. The practice shows that it is not always as efficient as it could be within a specific domain or in a real-world application, where GAs employing a more natural representation may outperform it. An extra problem with binary representations is that they present certain conceptual difficulties in mapping a real-world problem onto a bit string. For this reason, other structures have been used, for example the real or integer lists.

2.3.1.2. Initialising population

The initial population is mostly chosen at random, but it can also be chosen heuristically. This should be done carefully since GAs may quickly converge to a

local optimum if the initial population contains a few structures that are far superior to the rest of the population. The techniques used include perturbations of the output of a greedy algorithm, weighted random initialisations, and initialisation by perturbing the results of a human solution to the given problem.

2.3.1.3. Evaluation function

The evaluation function plays the role of the environment, rating solutions in terms of their 'fitness' and incorporate rule '*survival of the fittest*'. It evaluates members of the given population, and should have the maximum values at the optimal solutions. Sometimes, fine-tuning evaluation function is very important for GAs to obtain best results, so techniques such as scaling, normalisation et cetera are used.

2.3.1.4. Selection methods

The selection procedure is used to choose suitable candidates for mating. This is done by biasing the selection towards fitter candidates in the population. Selection has to be balanced with variation from crossover and mutation. This is the '*exploitation / exploration balance*'. If the selection is too strong then suboptimal high fit individuals will take over the population. As a result the needed diversity for further change and progress will be reduced. In a too weak selection will slow down the evolution. It will follow a description of the most common methods. However the relative literature does not provide guidelines for which method should be used for which problem [Mitchell 98].

2.3.1.4.1. Fitness proportionate selection

Roulette Wheel

A '*roulette wheel*' is divided up between each string in the population with the string holding a segment proportional in size to its fitness. The wheel is then spun and depending on where the ball lands the particular string holding that segment of the wheel is chosen and put forward for mating. This system works on the principle that the higher the fitness of a string, the bigger the segment of the wheel it holds and the more likely the ball will land in that segment. This process is repeated until the desired number of strings is obtained.

The main disadvantages of this method is that when the population settles down and individuals have similar fitness, the selection pressure decreases and then it doesn't

work any better than random selection. Also, negative fitness values tend to confuse selection process. Roulette wheel sampling is usually combined with other methods to determine which individuals will pass to the next generation.

Stochastic universal sampling

A drawback of the roulette wheel sampling scheme is that the probability distribution is sampled N times because the complete population is replaced. If the roulette wheel is called N independent times, this may result in a high variance in the number of offspring assigned to each individual. [Baker 87] developed an algorithm called *Stochastic Universal Sampling* (SUS) to reduce the variance. SUS also divides the roulette wheel into slices, but N equal spaced pointers are used to determine which individuals will get selected. The roulette wheel is then spun to determine the selected individuals. SUS is more efficient than roulette wheel sampling because it takes only a single pass to assign all the individuals [Mitchell 98].

The main problem associated with proportional selection is that it is scale sensitive. In the beginning of GAs, some individuals will have a fitness that is a lot larger than the fitness of the second best individual. This individual will take up a very large part of the roulette wheel, and will get selected many times. The other individuals do not really stand a chance to be represented in the mating pool. This can cause premature convergence towards a local suboptimum. In the last generations, all individuals are more or less equal in fitness. Each individual will receive an equal portion of the pie, and the probability to become selected will be uniform. The search capacity of GAs will reduce and the search behaviour will become a random walk. This can be overcome to some extent by using fitness scaling [Goldberg 89]. Fitness scaling rescales the fitness values between the maximum and minimum value, and modifies the probability distribution function. The disadvantage of this is that the probability distribution function is sensitive to the scaling parameters and that choosing different scaling parameters influences the selective behaviour of the operator even though the fitness of the solutions is the same.

Sigma scaling

Sigma scaling (like fitness reduction) is another method to address the problem mentioned at fitness proportional selection. When applying this method, the slice of the roulette wheel assigned to an individual is a function of the individual's fitness

value, the population mean and the population standard deviation. Sigma scaling performs much like fitness proportional selection with fitness reduction of 90%, but it consumes a little more CPU time [Mitchell 98].

Boltzmann selection

Sigma scaling keeps the selection pressure more constant over a run. But sometimes it may be good to have a stronger selection later in the evolution process in order to strongly emphasise highly fit individuals. One approach for this idea is Boltzmann selection [Mitchell 98].

Elitism

The elitist strategy [de Jong 75] simply ensures that the best or the n best strings in the population are taken into the next generation. The strings chosen are not crossed in any way, they are simply copied forward. In the literature appeared many suggestions from researchers that elitism significantly improves the GA's performance. This scheme is usually implemented along with another method.

2.3.1.4.2. Rank selection

The proportional selection schemes are easily influenced by so-called super-individuals at the beginning of the search process. At the same time are characterised by a weak search capacity at the end of the optimisation process unless fitness scaling is applied. In order to prevent these side-effects, rank-based selection procedures have been proposed. Ranking simplifies the mapping from the objective function to the fitness function [Grefenstette 97] and also eliminates the fitness scaling procedures, since selection pressure is maintained even if the objective function values within the population converge on a very narrow range. [Grefenstette 97] states that ranking may be a natural choice for problems in which it is difficult to state an objective function, for example if the objective function involves some subjective preference for alternative solutions. In that case, the exact value of the objective function is not so important. Additionally, [Michalewicz 99] states that rank-based selection operators control the selective pressure better than proportional schemes, and focus the search process better. On the other hand, these approaches have some apparent drawbacks [Michalewicz 99]. Firstly, it puts the responsibility to the user when to use these mechanisms. Secondly, rank-based procedures ignore the information about the

relative evaluations of different chromosomes and thirdly, all cases are treated equally regardless of the magnitude of the problem. Finally, selection procedures based on ranking violate the Schema Theorem [Michalewicz 99].

Tournament selection

In this selection method, k individuals (with replacement) are randomly picked from the population (k -tournament) at a time, and the one with the best fitness is selected. The larger the tournament size, the higher the selection pressure. For a population of size n , after performing n tournaments (or $n-1$, if elitism is used, since in that case the best solution will not have to compete for having a copy of itself in the following generation), the parent population for the next generation is complete. By the completion of the selection phase crossover and/or mutation is applied among the parent population for generation of offspring.

2.3.1.5. Recombination operations

After choosing the next population from the previous one, recombination operations are performed on them. The main operations are: crossover, mutation and inversion, but for many problems it is possible to define recombination operators that take problem specific knowledge into account. In typical GAs they used the three basic genetic operators: reproduction, crossover and mutation. Another operator rarely used, is the inversion operator. Description of some of the other (less-often used) operators (such as dominance, diploidy, duplication, deletion et cetera) can be found in [Goldberg 89].

2.3.1.5.1. Crossover

The most important operator in GAs is being responsible for most of the diversification occurring during the search process. During crossover, two randomly chosen parent chromosomes offer their parts to be swapped with a given probability for the creation of a new individual. Parent chromosome will receive an exact copy of itself only during the elite solution, without going through crossover. Crossover's crucial role is to interchange existing information between different solutions in order to generate new points in the solution space. The diversity of the initial population should be great in order to contain potential information from the entire search space. The common operators are one-point, two-point and uniform crossover. The

application of crossover is very depended of the length and structure of the chromosome representation. The survival rate of a fruitful schema is mainly high with the use of one-point and two point crossover, which is better from uniform crossover.

2.3.1.5.2. Mutation

It randomly changes an allele in a chromosome, with a given probability, to look for new points in the solution space. Mutation is thus an operator that acts very locally. Contrary to crossover it introduces new genetic information in the search that was not contained in the initial population. The decision for the mutation rate is based on the following rule of parametric studies: a mutation rate $m=1/n$, where n is the size of the chromosome, is almost optimal [Mühlenbein 97]. There are several mutation types that can be used. The common mutation type for the binary case where only one or two bits are flipped cannot be used here because the concept of complementary value is not defined. The following mutation types have been used:

Random mutation

For each variable that is going to be mutated, choose a random value within its range and assign this value to the variable. So, every value is possible.

Gauss mutation

This mutation is similar to the previous one, the only difference being that mutation step Δx is calculated according to Gauss' distribution $N(0, 1)$: smaller mutation steps are much more probable than large mutation steps. This is a standard Evolutionary strategies mutation [Back 97].

Exponential mutation

This mutation type comes from the idea that the role of mutation at the beginning is to make large jumps whereas later on, as the search progresses it should be used more for fine-tuning so small jumps are more desirable. Here c is the constant that depends on the generation number.

2.3.1.5.3. Inversion

This operator inverts the sequence between two randomly assigned points in a single chromosome string. Inversion alone has no immediate effect on string fitness. If the current population contains bad ordering, there is a high probability that crossover

will destroy this schema. Inversion lowers the probability of destroying it. Goldberg provides some theoretic analysis of inversion, [Goldberg 89]. Inversion is not used very much nowadays.

2.3.1.6. Population size

The population size used and the type of search progression are two central questions for typical GAs. The guidelines for setting population sizes are very poor, for a typical GAs experiment. Those settings are problem dependent and they must be fine tuned by the user. A better convergence is achieved with large population sizes eventually lead due to the larger pool of schemata available. Unfortunately such populations have an initial large inertia and so a poorer initial performance. A large number of generations are needed in order to converge to high-performance solutions, so too large population sizes tend to slow down the algorithm. Moderate size populations may be a reasonable compromise between finding good solutions and speed. Earlier studies have shown that typical population sizes for GAs range from 30 to 200, [de Jong 75].

2.3.1.7. Advanced Genetic Algorithms

Many problems do arise when applying GAs to highly complex applications. The most common is premature convergence where the population converges early onto non-optimal local minima [Davis 91]. Problems are also caused by deceptive functions, which are, by definition, ‘*hard*’ for most GAs to solve. In addition, noisy functions [Goldberg 92], and the optimisation of multiple criteria within GAs can cause difficulties [Fonseca 95]. In an attempt to overcome such problems, new, more advanced types of GA are being developed. These include: Parallel GAs, where multiple processors are used in parallel to run the GA [Levine 94]. Distributed GAs, where multiple populations are separately evolved with few interactions between them [Mühlenbein 92]. GAs where applied niching and speciation techniques [Horn 94]. Messy GAs which they use variable-length chromosomes and a two-stage evolution process [Deb 91]. Multiobjective GAs allows multiple objectives to be optimised with GAs [Srinivas 95], [Bentley 96], [Coello 96]. Structured GAs where parts of chromosomes are allowed to be switched on and off using evolvable ‘control genes’ [Dasgupta 97], [Parmee 94].

2.3.2. Multi-objective Optimisation

In many design problems the simultaneous satisfaction of several objectives is a necessity, (minimise cost and maximise performance at the same time). Multi-objective optimisation (multi-criteria optimisation, multi-performance or vector optimisation) can be defined as the problem of finding a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These functions define a mathematical description of performance criteria which are usually in conflict with each other. Hence the term ‘*optimise*’ means finding such a solution which would give to of all objective functions values acceptable to the designer [Coello 96]. The numerical quantities for which values are to be chosen in an optimisation problem are called decision variables. In the design problems any imposed restrictions are called constraints. Constraints must be satisfied in order to consider that a certain solution is suitable. Constraints are restrictions that describe dependences among decision variables and constants or parameters, involved in the problem. These constraints will be expressed in form of mathematical inequalities and equalities. The number of equality constraints, must be less than the number of decision variables. Otherwise the problem it could be over-constrained since there are no degrees of freedom left for optimising. The evaluation criteria are expressed as computable functions of the decision variables that are called objective functions. In many cases, some of them will be in conflict with others, and some will have to be minimised while others are maximised. Objective functions may be commensurable (measured in the same units), or non-commensurable (measured in different units).

It should be noted that the terms ‘*multi-criteria*’ and ‘*multi-objective*’ are often synonymous in the literature. However, in this work, as suggested in [Coello 02a], the expression ‘*multi-objective optimisation*’ will refer solely to the presence of multiple objectives, while ‘*multi-criteria optimisation*’ will imply the use of an additional procedure to deal with the user’s preferences. Multi-objective methods combined with evolutionary algorithms [Zitzler 99] are briefly presented below, before focusing on a priori techniques. The solution in general is not unique in vector optimisation. Therefore the user has to provide additional information about personal preferences in order to find the optimum solution. The following three different approaches are available in the literature [van Veldhuizen 98], [van Veldhuizen 99], [Zitzler 00]:

2.3.2.1. A posteriori methods

Preferences may be used at the end, when the Pareto front has been completely determined. In a posteriori methods, the main step consists in drawing up the shape of the Pareto front. A posteriori techniques aim to determine the shape of the whole Pareto front, and let the user decide which solution to retain. There are various methods to find out the non-dominated solutions using evolutionary algorithms [Coello 02a], [van Veldhuizen 00].

2.3.2.2. Progressive methods

Preferences may be used during the optimisation process, in an interactive way. Though some methods have been developed since the 1970's to use information from the user within the search process, like the Surrogate Worth Trade-off (SWT), or more recently Jahn's, Geoffrion's, Fandel's [Celleste 02] or Tappeta's [Tappeta 99] methods, Coello [Coello 02] underlined that there is extremely few works dealing with interactive methods implemented in Evolutionary Algorithms (EAs), since they require an important investment of time from the decision maker.

2.3.2.3. A priori methods

Preferences may be included since the beginning of the search process (a priori methods): the user has to assign a weight to each criterion, or at least a ranking of the m objectives. In *a priori* methods preferences may be included since the very start of the search process, by a ranking of the objectives, or more commonly through weights assigned by the decision maker to each criterion. This can be very useful in particular when the user already has a strong idea about his/her preferences about the objectives, or when the number of objectives exceeds three (which makes difficult and less intuitive the choice between the non-dominated solutions). As a matter of fact, even with 2 or 3 objectives, making a choice after the determination of the trade-off surface is a complex task, generally requiring a preliminary treatment of the non-dominated solutions, as the filtering indicated above. Coello also noticed that there is very little work in which the preferences are explicitly handled in the evolutionary multi-objective literature [Coello 00b]. Therefore, in the scope of this thesis, the emphasis is put on using preferences since the beginning of the search process. In general it is difficult a decision on which method is better. Therefore the method selected by the

user must be chosen with respect to his/her needs. Here are the most popular a priori methods, and some of their implementations in EAs [Fonseca 95], [Horn 97]:

Lexicographic Ordering: the user must rank the objectives following their relative importance [Coello 02a]. In this method no weights is used. Beginning with the most important one and proceeding according to the predefined order of importance the optimum is found by minimising the objective functions. The main limitation of this approach is that when the number of objectives is high, it tends to optimise the most important ones. Furthermore, no quantitative preferences can be added to the process. [Coello 02a].

Weighted Sum Method: the most popular a priori method in the EAs is the weighted sum, and moreover among design engineers. The m objective functions are aggregated into one, [Osyczka 02]:

$$f(x) = \sum_{i=1}^m w_i f_i(x)$$

The weights w are as following:

$$\sum_{i=1}^m w_i = 1$$

A collection of applications of linear aggregation of objectives techniques implemented in EAs is available in [Coello 02a].

2.3.3. Handle constraints in Evolutionary Algorithms

Constraint handling in Evolutionary Algorithms (EAs) is neither simple nor straightforward. Especially this happens in design problems where applications are normally characterised by many constraints in the form of physical, technical or economical requirements. Handling of constraints for single-objective problems are the following [Michalewicz 95], [Coello 99]:

2.3.3.1. Death penalty

This is the most straightforward way to take the constraints into account. In the random creation of the initial population, and during the selection of the best individuals at each generation, the unfeasible solutions are systematically eliminated. The main drawback of this approach is that all the unfeasible solutions share the same fitness value, so no useful information about the unfeasible domain is exploited.

2.3.3.2. Penalisation methods

An individual might be penalised because it is unfeasible. This method does not take into account the amount of violation. Penalty methods are: Static penalty, Dynamic penalties, Annealing penalties, Adaptive penalties [Coello 02]. Penalisation techniques provide good results without significant modification of the standard evolutionary algorithm. However, their main drawback appeared in the difficulty in the choice of the parameters. The reason is that no general rule can be applied to determine their values.

2.3.3.3. Decoders

Decoders offer in general an efficient alternative to classical penalty-based methods. Indeed, they provide a decoding process, which can build automatically feasible solutions, following instructions stored in the chromosomes [Back 97].

2.3.3.4. Repair strategy

The principle of Repair algorithms is to transform an unfeasible chromosome into an admissible one, thanks to knowledge about the problem [Michalewicz 96].

2.3.3.5. Constraint-preserving operators

Genetic operators of GAs (like crossover and mutation) can be modified to maintain the feasibility of the population [Back 97]. For example, if equality and inequality constraints are linear, and if each variable set is uniform, then boundary and non-uniform mutations and arithmetical crossover automatically transform feasible parent(s) into offspring automatically satisfying the constraints.

2.3.3.6. Separation of Objectives and constraints

An alternative approach is the separation of objectives and constraints. All methods making a distinction between objective(s) and constraints are classified in four approaches: co-evolutionary algorithms, superiority of feasible points, behavioural memory, and multi-objective optimisation techniques [Coello 02].

2.3.4. Choice of weights

The definition of importance weights is an intuitively process. In addition in many design applications importance definition is very tremendous and time consuming. In

the literature appeared some approaches to aid the user in reflecting appropriately his/her preferences [Cvetkovic 00], [Ezzegaf 00].

2.3.5. Evolutionary Design

Evolutionary design is an approach that utilises different evolutionary computation techniques in various different design domains. The strength of evolutionary design comes from the observation that controlled evolution can be formulated as a general purposed problem solver with the ability similar to human design intelligence but with magnitude of speed and efficiency. Traditional AI methods such as ‘*rule-based reasoning*’ have to model design intelligence explicitly in terms of knowledge both in representation and inference. These methods have serious drawbacks because the process of how human designers actually use this kind of knowledge is not necessarily fully understood. Bentley [Bentley 99] gives a classification of Evolutionary design into the following categories:

- Evolutionary design optimisation. Optimisation of existing designs by evolving the values of suitably constrained design parameters;
- Creative evolutionary design. Generation of entirely new designs from little abstract knowledge to satisfy functional requirements;
- Conceptual evolutionary design. Production of high level conceptual frameworks of preliminary designs; and
- Generative evolutionary design. Production of forms of designs contributing to the emergence of implicit design concepts.

These evolutionary design approaches combine several vital aspects of design intelligence in an evolutionary process including modelling design data and information, concept formation, idea generation, optimisation, learning, and evaluation. As Evolutionary design extends and combines CAD with analysis software, it provides excellent potential for developing more intelligent design support tools.

2.3.6. Genetic algorithm applications in design

One of the regions in which GAs perform well is design optimisation and many results have been reported in the last ten or fifteen years [Holland 92] and [Goldberg 94]. Different problems in different areas have had solutions successfully optimised

by GAs and design problem is one of common problem areas to use GAs as a way of optimisation. GAs are being applied to many areas of engineering design in mechanical engineering, electrical engineering, aerospace engineering, architecture and civil engineering, et cetera. It is practically impossible to give a comprehensive overview of all existing applications even for one such area. We have decided to discuss conceptual design, and shape optimisation. The common feature of these areas is their strong geometric nature, which is also important in most design problems. This also indicates that GAs can be efficient in solving problems with very different engineering content within a similar framework and by using similar procedures.

2.3.6.1. Conceptual design

Conceptual design of an artefact takes place in an early stage of design and usually requires the designer to act creatively. The designer either uses novel components or combines known components in a novel way. The design parameters to be optimised are decided at this stage of the design process. There could be several ways of constructing good conceptual designs, but there is no fixed methodology to follow. In many studies have shown adaptive search techniques with emergent solution characteristics provide a computing paradigm that is well suited to the complicated and unstructured nature of the conceptual design process [Grierson 97].

Goldberg [Goldberg 91] presents an idealised framework for conceptual design. The essence of conceptual design is captured in four components: a problem to solve (the design challenge), someone to solve it (the designer), one or more (conceptual) designs, and a means for comparing alternative designs (the design competition). He shows how GAs can be thought of as '*a lower bound on the performance of a designer that uses recombinative and selective processes*'. In other words, a human designer should perform a conceptual design task at least as well as GAs. This literature review focuses on research concerned with conceptual design of buildings, bridges, spatial planning, product design, and airframe design. It is important to note all areas of research are under continual development. Furthermore, it is necessary to mention that the researches discussed in the following do not cover all aspects involved in the global conceptual design process but do try to address the problem from several important viewpoints.

Frazer used GAs in his evolutionary architectural design to evolve unpredicted forms of architectures and their possible interactions with the environments [Frazer 95], [Frazer 99], [Frazer 01].

Chakrabarti developed a functional synthesis program that generates a large number of abstract design concepts from functional requirements and abstract building blocks of engineering elements [Chakrabarti 96].

Jo and Gero [Jo 96] employed a GA for space layout planning. In the study they optimise the distribution of available space among different activities in a building in order to minimise the cost of taxiing between those activities. They concluded that a GA is able to generate good designs for complex design problems.

Bentley [Bentley 97] describes a generic evolutionary design of solid objects using a GA system that evolves new conceptual designs from scratch. Conceptual designs of three-dimensional solid objects were created and optimised. Multiobjective optimisation is investigated to allow users to define design problems without fine-tuning large numbers of weights. A variable-length chromosome in GAs is addressed to allow the number of primitive shapes that define a design to be variable. This problem is overcome by the use of a new hierarchical crossover operator, which uses the new concept of a semantic hierarchy to reference chromosomes. The demonstrated system was very feasible in the evolution of both conventional and unconventional designs for many different solid-object design tasks.

Park and Grierson [Park 98] developed an algorithm for the optimal conceptual design of medium-rise buildings accounting for the cost of the structure and the quality of occupant space. The approach generates best-concept designs by simultaneously optimising two conflicting criteria concerning the project cost and the flexibility of floor space usage. Specifically, Pareto optimal equal-rank designs that are not dominated in both criteria by any other feasible design are found using a multi-criteria GA. They found that there is a performance trade-off between the objective criteria and that it is up to the designer to make some compromises to arrive at an acceptable design.

Gero and Kazakov [Gero 00] use GAs to enlarge the state space, so that the set of possible designs changes. They generalise crossover in such a way that it can move the population outside the original state space. This strategy supports creative design, and therefore can be used in the conceptual stage of the design.

Shrestha and Ghaboussi [Shrestha 98] discussed a methodology for the evolution of optimum structural shapes in which a GA is used to evolve optimum shape designs that are free to assume any geometry and topology and do not necessarily resemble any conventional design. The methodology addresses configurationally and topological aspects of the design, and considers discrete member sizes and multiple loading cases for planar and space structures.

O'Reilly [O'Reilly 98] applies GAs to the design of three-dimensional shapes, to help designers at early stages of design mainly by proposing unexpectedly innovative shapes. The difficulty of finding appropriate objective functions for design evaluation have made some of these attempts remains at an experimental level, with selection and crossover being performed manually by the designer.

In Cvetkovitc [Cvetkovitc 00] is explored the problem of conceptual engineering design and the possible use of adaptive search techniques and other machine based methods. For the *multi-objective optimisation* (MOO) within conceptual design problem, GAs adapted to MOO are used. Decision support methods within conceptual engineering design framework are discussed and a new preference method developed. Interactive dynamical constraints in the form of design scenarios are introduced. The use of machine-based agents in conceptual design process is investigated. They are integrated with the conceptual engineering design system to form a closed loop system that includes both computer and designer.

Galdas [Galdas 01] has suggested a GA Generative System (GS) for the environmental performance -behaviour of buildings in relation with the study of more particular problems such as building envelope design. The GS is further used to generate whole building geometries, departing from abstract relationships between design elements and using adaptation to evolve architectural form. In that GS system the idea of '*optimisation*' was excluded (trying to find the best possible solution), and instead is providing many different, unexpected options, which might not be '*optimal*' but would nevertheless have a good environmental performance and be architecturally interesting at the same time.

Sun [Sun 01] has applied generative and evolutionary techniques in Computer Aided Product Design (CAPD). A computer model of a generative evolutionary product design process for conceptual design was achieved through the application of GAs. The system is integrated with a commercial CAD system for 3D visualisation. A multi-objective evaluation and selection are investigated to allow users to specify

initial design requirements in terms of Design for Manufacturing (DFM) considerations and constraints. The integration of DFM constraints into the database and the evaluation step of the GA provided insights on how general DFM issues can be incorporated at the initial stages of conceptual design.

2.3.6.2. Shape optimisation

Shape is one of the most important characteristics of technical objects. Their compliance to functional requirements and production costs strongly depend on shape. Considerable efforts are continuously exerted in engineering science to find better shapes, or to optimise the shape of a component subject to engineering constraints. While classical methods for optimisation often fail under such complicated conditions, GAs may offer solutions in many practical situations. In addition to this, design constraints that are very hard to handle by analytic methods can be directly incorporated into a genetic optimisation. There is a great variety in the number and structure of shape parameters of different technical problems. Among them are problems of truss and bridge design [Moore 97], [Park 98], [Shrestha 98], [Khajehpour 01]. Shapes with complex geometrical properties can be represented by two or three-dimensional grids of control points of Bezier, B-spline, NURBS surfaces or volumes [Renner 03].

2.3.7. Discussion

The generation of concepts whose representation is of unknown size and shape is a very difficult task. An example of such a concept is the creation of spatial forms. Composing the three-dimensional spatial structure of a building-space is one of the most important open tasks in architecture. Architects work in ways that imply formulation of many items of information which have varying importance to their designs. They recognise their applicability, while working at the details of the solution. During the design process, it is not possible to start out by making lists of criteria that are supposed to get satisfied with the newly created layout. The whole process for a good solution is also a search for proper information with which to evaluate it [Rychener 88]. As we saw from the above analysis GAs embody a range of dynamics that permit task-specific knowledge to emerge while solving a given problem and they are better suited for global search and global optimisation in large and complex search spaces than the traditional exhaustive search algorithms such as

breadth-first search or depth-first search. Considering that each generated solution is evaluated according to the user's aesthetic criteria, one can view the problem of finding the optimal solution as a global optimisation problem that searches for the maximum of the evaluation function. Typically, these are *NP*-complete problems and one is satisfied by a 'good' solution rather than the 'optimal' one. Using a population of solutions is thus intrinsic to the mechanics of the algorithm, apart from being a beneficial advantage in terms of its solutions. Although the solution spaces do present global minima as well as local ones, they tend to be quite flat around the global minima. Usually, the solution spaces found in an architectural design problem are not characterised by the presence of spikes (that is, having very good solutions surrounded by very poor ones). This means that within some range, the relative impact of changing some of the variables is small. Outside of this tolerance range, however, the impact in objective function values can be quite significant. This can provide useful information to the architect about the more sensitive points in a design, and on the consequences of introducing certain changes in relation to the best solution found by the algorithm. Because this system was conceived as an open-ended tool, one with which the architect-designer could engage in an interactive loop, by running the program, getting some results, introducing some changes, and running the system again, this was considered to be an important advantage towards the use of GAs. Another criterion for using a GA was that GAs uses a coding of the variables during the search process, not the variables themselves. The chromosomes can then be decoded into any kind of variable, numerical or non-numerical, what provides an extra degree of flexibility towards using the system for architecture applications. For example, part of a chromosome can be decoded into a material name, or into any other kind of non-numerical variable.

2.3.7.1. Why Evolutionary and Adaptive Computing?

The evolutionary and adapting techniques can offer significant advantages over more traditional optimisation techniques. The common attributes of the various evolutionary and adaptive search techniques of particular relevance to practical, complex problem-solving include:

- Requirement for little, if any, a priori knowledge relating to the problem. Evolutionary and adaptive search techniques can therefore utilise a wide range of model / simulation type and structure

- Excellent exploratory capabilities. The techniques initially randomly generate trial solutions and the extent of subsequent search around such solutions depends upon their relative performance. Further sampling of diverse solution sets can continue throughout the search process through the action of various operators.
- Ability to avoid local optima. The stochastic nature of the various algorithms combined with continuing random sampling of the search space can prevent convergence upon local sub-optima.
- Ability to handle high dimensionality. Successful application to problems described by greater than four hundred variable parameters is possible.
- Robustness across a wide range of problem class. The techniques can generally outperform more deterministic optimisation algorithms across a wider range of problem classes.
- Provision of multiple good solutions. If required evolutionary and adaptive search strategies can be developed that identify multiple high-performance solutions.
- Introduction of multi-objective approaches, which can be easily and successfully integrated to provide a range of high-performance compromise solutions for further off-line evaluation.

Evolutionary and adapting techniques include evolution strategies, GAs, evolutionary programming, genetic programming, ant colony models, tabu search, and simulated annealing. These techniques have become firmly established and are now becoming widely utilised within industry. Further, it will be seen that GAs differs from Constraint Satisfaction Problem (CSP) applications – approaches to the architectural design problem. Other researchers have showed that GAs differ considerably from other methods described there, like Simulated Annealing and Tabu Search, in that it searches from a population of points instead of a single point [Galdas 01].

2.3.7.2. Justification of Choice Using Genetic Algorithms

The main criteria for choosing GAs for the search and optimisation module was the fact that they provide a population of final solutions instead of a single one. This represented a move towards the *declarative modelling - adaptation paradigms* that were in the genesis of the work, and were also considered to be an important factor in an application area like early phase architectural design. A design system that would

provide a single '*optimised*' solution might have poor acceptance among architects, who might prefer to have a range of choices between different solutions, all having a high standard of performance according to the (multi)-objective functions included in the search, and from where the architect could exert further choice, according to other criteria like personal aesthetic preference, initial design intentions, et cetera. It is important that the evolutionary declarative design system described in this thesis requires an algorithm capable of consistently finding improved solutions to a collection of architectural design problems, i.e. the evaluation criteria specifying new design tasks for the system could consist of literally any type of function. Furthermore the search for the morphology of a building is a very complex and tedious task. For this reason it is felt that GAs is an appropriate choice to form the core search-generation engine of the declarative modelling system.

2.4. Architectural Design

In section three we present literature review from Building modelling and architecture knowledge. In architecture the '*problem – solving*' procedure concerns design activities. These activities could characterise as highly creative. In architectural design it does not exist any predetermined strategies to achieve a set of goals [Tzonis 94]. In general [Simon 96] the design process is characterised by:

- Complexity
- Fuzziness
- High degree of subjectivity.

We could define architectural design as: The process of applying various techniques and principles for the purpose of defining a spatial system in sufficient detail to permit its physical realisation. For any architectural product or system, the very first step of the development phase is design with the goal of producing a model or representation of an entity that will later be constructed. The process deals with the combination of intuition and judgment based on experience building similar models, a set of principles that guide the way in which the model evolves, a set of criteria that enables quality to be judged, and a process of iteration that leads to a final design representation.

It is very important that an understanding of the requirements emerges as the design activity proceeds, and that is why design requirements are generally '*ill-defined*' and

often contradictory [Simon 96]. *Design problems* are often called ‘*wicked*’ or ‘*ill-defined*’ problems. Characteristics of these problems are given in [Cross 94]. The most well-known characterisation of design problems in the field is given in [Lawson 90]:

- Design problems cannot be comprehensively stated.
- Design problems require subjective interpretation.
- Design problems tend to be organised hierarchically.

More characteristics of design problems can be found in [Evbuomwan 96]. However, most design procedures are indirect, in that a design concept is proposed and then successively analysed, evaluated, corrected and reanalysed until the final results fulfil the designers’ demands. The success of such a design process depends very much on the initial design concept proposed and on the opinions, judgements and experience of the designers. As such, the corresponding design process is often relatively ineffective since the structural type and arrangement, architectural layout and electrical/mechanical equipment are often simply devised and copied from previous designs.

Designing is a dynamic process. The compositional methods incorporate the transformation of familiar metaphors and project them into new situations. The building design process has as a basic objective to develop, design and construct a building to fulfil a given set of performance requirements, building regulations, state urban codes, user requirements, designer/user aesthetic preferences, et cetera [Lawson 94], [Rowe 91].

Design knowledge is in general *implicit knowledge* that obtained through experience or *explicit knowledge* that concerns design theories, methods, strategies, design patterns, types of designs, and earlier design projects. Design knowledge could be analysed in two categories [Lawson 94], [Rowe 91]. The first category concerns rational knowledge that includes design codes and function analysis. The second category concerns spatial composition, space planning and architectural style.

During the design process, architects based their research also on the knowledge that defines an architectural style. Architectural styles have a twofold role in design. First are systems that enable aesthetically pleasing proposals. Secondly, they provide means to tackle complexity in design [Broadbent 90]. While the existed styles combined and reassembled there is a control and an optimisation of the given

functional demands. When the latter are not satisfied in the designs then rational knowledge is introduced to direct further the process of design. In a design process like this it is possible the inheritance of some *functional* and *formal-typological* fulfilment from precedents design proposals. A phase of design uses existed design knowledge in a particular and sometimes innovative manner, for the production of new design proposals. Therefore design knowledge could extract from an ensemble of design objects. A very important dimension of design knowledge is style that is expressed by particular features that belong to a class of objects created by a single person or a group during a specific period of time.

Some of the objective criteria in this multi-level process are entirely rational and quantitative, but others must remain non-quantifiable because of either their enormous analytical complexity or because they involve elements of taste or aesthetics. The fundamental aim in dealing with objective criteria is to find optimal solution to the problem at hand. In a building optimisation problem the, optimal solutions are those that satisfy the requirements of function and behaviour while remaining within the aesthetic limits forced by the architect. This research will focus in particular upon identifying ‘*optimal concept*’ designs. Equally significant will be its focus on the development of a general approach by which such designs may be achieved.

2.4.1. Hard – Soft constraints

The design of a building involves the development of the physical description of an artefact subject to a set of given constraints and specifications. Architectural design is guided by the constraints on the spatial composition and the morphology of the final building. A candidate solution can be considered satisfactory if it meets and respects the user demands and requirements. These properties are formalised through a number of constraints. Constraints can be of two types: *hard* or *soft*, depending on whether their satisfaction is mandatory for *hard* constraints or just desirable in the case of *soft* constraints.

In a given building description, the initial building requirements are considered as *hard* constraints. On the other hand, any stylistic principles for the design of the specific building are considered as *soft* constraints. In order for a building to belong to a particular style, specific principles in the form of several architectural constraints have to be respected. The constraints are interpreted to restrictions on the existence or

not of specified objects, properties and relations in the building design. Such constraints are characterised as soft constraints.

The initial description of a building by the designer (i.e. the brief) represents a set of objects under a variety of hard constraints and soft constraints. Hard Constraints differ from soft constraints in that *soft* constraints must be satisfied, while *hard* constraints must not be violated. Given the brief of a building project the designer defines specific demands that will take the form of *hard* constraints that the design of the building would not violate. In parallel, there exist a number of *soft* constraints directing the final design of the building. An example of a *hard* constraint would be: '*two spaces are adjacent on specific sides*'. If, for a particular style, spaces should be coaxial (i.e. their centres are on an axis that passes through the midpoint of their sides) then a *soft* constraint could be: '*two adjacent spaces should be coaxial*'. Such a requirement could be satisfied as much as possible in order for the resulted building design to belong to the particular style. This *soft* constraint does not say that there should be a *coaxial* relation, but that if there is for a building to belong to a particular style, it should be satisfied as possible. A solution will be said to be feasible if it satisfies all the *hard* constraints, and a feasible solution will be said to be more or less acceptable depending on the degree to which *soft* constraints are satisfied. Hard constraints, to which the product or building *must* conform, act differently from and soft constraints (guidelines, style), to which the product *should* conform. An important thing to notice is that the classification of each constraint as *hard* or *soft* is to some extent arbitrary, and may be thought as being part of the problem instance definition process. However, in logical terms the difference amounts to the distinction between logical consequence (for *soft* constraints) and logical consistency (for *hard* constraints).

Both *hard* and *soft* constraints can be *local*, i.e. applicable to a single parameter, a single object or to a relation between two objects, or a constraint can be *global*. For example the hard constraint that '*the length of a building must not exceed 20 meters*' is a global constraint since the length of a building is not a property of any of the spaces. On the other hand, a soft constraint like '*space A should be longer than wider*' is a local constraint between a pair of properties of a space.

When designers are unable to create designs conforming to all the *soft* constraints, they weaken or discard the less important constraints, to make their designs produced by their standard methods meet the task demands as well as possible. But when *hard*

constraints are in conflict, they can ensure that no standard design will work. This situation forces designers to try to innovate, by exploring and using reflective problem solving strategies, and progressively refining their understanding of the problem. From repeated failures and partial successes they refine their strategies for reformulating problems and generating novel ideas.

2.4.2. Architectural building design phases

Building design process is a systematic procedure used by engineers, (in the beginning by architects, and by other categories of engineers), and designers to create structures to meet the needs and desires of the users/inhabitants. This procedure is used to prepare a set of plans and specifications that guide contractors, in the completion of their work. During the design process it is possible a transition without interruption between major phases by reusing information from each stage to the next. The number of steps in the process will vary from country to country according local building guidelines, or scientific institutions (the American Institute of Architects – the Royal Society of British Architect – the Maîtrise des Ouvrages Publics). We consider the building design as a four stages process [Sen 98], [Pahl 96].

2.4.2.1. Conceptual design

It is that part of the design process in which, by the identification of the essential problems through abstraction, by the establishment of function structures and by the search for appropriate working principles and their combination, the basic solution path is laid down through the elaboration of a solution principle. Conceptual design *determines the principle* of a solution [Pahl 96]. Conceptual design is the earliest phase of the building design process and commences with a set of initial concepts. Designers must evaluate different competing criteria with the view to achieve a good compromise design. It is very important to underline the fact that there is no single solution with optimal performance with respect to all requirements due to the fact there are conflicting objective criteria. That is, the selection of a suitable solution involves making subjective compromises between conflicting objective criteria. The conceptual design phase involves making decisions that can have the maximum influence on the final design and project cost. One study suggested that as much as 80% of the total resources required to construct a building are committed by the decisions made in the first 10% of the design process [Deiman 93]. Albeit, designers

often tend to spend most of their working time on the detailed design phase, where the scope for significant improvement is much less. They often only generate a single design concept, or at most a few that satisfy the design criteria, because traditional design practice places severe constraints on time and design costs.

Extensive generation and evaluation of alternatives is only possible with the help of computer-based methods. Those computer methods for conceptual design are not yet available to designers in practice. One reason for this situation is that conceptual design has not yet evolved into a well-defined procedure. An overall view of the design process and the design itself is needed when performing conceptual design. The designer at the early stages must understand the many factors affecting the building being designed. Such a global approach to building design should include account for engineering requirements, aesthetic preferences, quality of space and comfort, and finally financial (operating cost, and rental revenue). Significant complexity comes from the need to determine the relative benefits of all of these various quantities and qualities [Rush 86].

2.4.2.2. Preliminary design

Some authors do not consider preliminary design as a separate stage and consider it a part of conceptual design. The preliminary design stage involves the initial development of one or a few conceptual models. According to Dym [Dym 94] the preliminary layout is obtained by refining the conceptual designs and ranking them against the design specifications, and choosing the best as the preliminary design.

2.4.2.3. Embodiment design

It is that part of design process in which, starting from the working structure or concept of a technical product, the design is developed, in accordance with technical and economic criteria and in the light of further information, to the point where subsequent detail design can lead directly to production [Pahl 96].

2.4.2.4. Detail design

The detailed design phase defines a complete solution for all subsystems, and results in final drawings for architectural, structural, electrical and mechanical systems. It is that part of the design process which completes the embodiment of technical products with final instructions about the layout, forms, dimensions and surface properties of

all individual components, the definitive selection of materials and a final scrutiny of the production methods, operating procedures and costs [Pahl 96].

2.4.2.5. Problems in conceptual design

Our main focus is on the conceptual phase of architectural design. Some of the basic problems of conceptual design [Cvetkovic 98], [Cvetkovic 01], are briefly explained as following.

In a given problem there exist constraints and objectives but their distinction is very often very fuzzy. In many cases examination and understanding of the problem ends up with a move from objectives to constraints or vice versa.

The nature of constraints could be either hard or soft. The continuous expanding of problem demands some of them could change character from hard to soft and vice versa. It is ever possible the elimination of some of constraints during the process.

In many design problems exist always great number of variables. Moreover their values have fuzzy and flexible ranges. Additionally, the limits of values are not always known from the beginning and could be rather artificial boundaries.

Conceptual design could ends towards the achievement of optimal solutions that provide knowledge for the extension of ranges, the inclusion of alternative constraints, and/or the removal of some of them.

Off-line analysis is required in many cases, and therefore a set of solutions should be able to be saved in a database for easy visualisation and reuse. Furthermore, it should be able to be entered in other programs for alternative evaluation of given solutions.

An exhaustive search of the solution space is not needed in architectural design because it could be time consuming and demand from the designer an evaluation of enormous solutions. When an optimisation program is used the number of parameters should not confuse the designer. Additionally the possibilities offered by the optimisation program should not perplex the design process.

Interaction with the search process is highly aid the designer. The sampling of results after a number of evaluations enables further adapting of design parameters and constraints.

The fuzzy nature of initial design concepts and the many different variants that architects wish to try are problems of conceptual design. Computers should be able to help them in exploration of those variants whilst suggesting some others as well.

2.4.3. Architectural Design knowledge modelling

Architecture-Engineer-Construction (AEC) domain includes numerous efforts to develop computational environments, for an ambitious development of buildings. In CAAD appeared a great number of building product models. Different specifications of them were developed by national or international organisations as industry standards. Numerous commercial design programs have provided tools for introduction of building models during the design process. While they provide functions, during the building's design phases, they are quite far from introducing a model to deal with the complexity and open-ended character of architectural design. Architectural design knowledge is a particularly challenging information domain to model. Especially the representation of the early phase of architectural design is one of great richness. During that phase, architects have the role of designer that conceive an idea of a product and use *visual* and *verbal* descriptions in order to express it. In an architectural design system, this expression is interpreted to design with the goal of producing a specific representation of an entity that will later be constructed. Therefore, design encompasses the development of building's form through visual, factual representations and verbal descriptions. CAAD needs to be understood as a '*dialogue*' between architect and the CAAD system. However, what was lacking in most of the CAAD software was such an environment.

The objectives of modelling architectural design information are to enhance representation and presentation of information, to formalise knowledge reasoning in order to aid the design process and to automate parts of the process, to improve communication of information between different participants in building process, to improve quality control, to enhance the design process and to enable an easier management of design and construction project.

The modelling of architectural design information has the following requirements in order to define and structure such information. First, to confront the problem-solving in design in terms of both finding a proper solution, and identify the design problem itself. Second, to confront the evolutionary development in design processes where information is not static but subject to change. Third, to deal with the evolution of designers' approaches to design problems.

In the AEC domain the developments of computer based information systems enables a computer integrated construction process, where information is generated, used and

communicated between different actors in planning, design, production, use and maintenance of the built environment. In a computer integrated construction process the following prerequisites are considered as important, [Ekholm 98].

- Information must be structured into computer based models in order to enable computer based analyses of the products and processes that are developed,
- The computer must be able to handle information of other objects than buildings, e.g. the user organisation, the site, the construction process, and the facility management process,
- Information must be standardised in order to be consistent throughout the processes,
- Information must be computer based already in the initial processes,
- It must be possible to use the computer as a design tool.

There were many developments of computer based information systems in order to handle the rich information of the building process. In general these approaches are distinguished in to categories *static* and *dynamic* modelling.

2.4.3.1. Building modelling

The automatic generation of necessary information for the realisation of a building project is the ultimate goal of *Architecture-Engineering-Construction (AEC)* domain. All information relative to a building project has to have a structure. Structured representation of such information it is very important mainly among the different phases of the design process. A branch of construction Information Technology is *Building Product Modelling (BPM)*. The main effort of BPM is the creation of computer systems in order to gather all relevant information about a building for purposes of design, production and maintenance. The fundamental idea is the employment of a common building product model which could be used and manipulated by everyone that play a role in the design – construction process, from initial design to maintaining process. A building product model could be defined as a digital information structure that incorporates the objects making up a building, and captures the form, behaviour and relations of the parts and assemblies within the buildings [Eastman 99]. According to this, a building model introduces many representations of all the life stages of a building, while addressing model issues from

different engineer domains. In general, the denotation of product modelling refers to specifying the data models to represent products of a certain kind. The term product data refers to creating a model of a specific product, using such a data model [Björk 95]. The modelling paradigm of *meta-level* provides support for all data modelling. This level enables the creation of explicit concept definitions. Examples of such paradigms are Relational databases, Entity-Relationship Models, Frames, and Object-Oriented modelling. In particular product data modelling finds *concepts* used in connection with the kind of product to be modelled. These concepts are represented following by data schemes as defined by the modelling method. These concepts are distinguished in three categories, *entities* of material things, *relations* between things, and intrinsic *properties* of things. In product data modelling appeared two approaches in defining concepts. The first approach concerns *single-aspect* view of modelling domain-specific concepts. The second approach is *multi-aspectual* modelling.

In the *AEC* community appeared number of efforts to develop information models of building models. In general, different developments approach building design mainly from the scope of product information. The goal that achieved was the development of common representations of complex products for communicating information between *CAD* systems. On an international level standardised representations are confronted by the work of *STEP* (*STandard for the Exchange of Product information*) [ISO 92], [ISO 93] through ISO 10303 (*International standard for the computer-interpretable representation and exchange of product data*). The PART 225 is a model that concerns the geometric aspect of building elements and facilitates the exchange of geometric data [ISO 96]. In parallel, the *IAI* (*International Alliance for Interoperability*) activity provides the IFC (*Industry Foundation Classes*) as a collection of object classes on building industry [IFC 01].

In construction information research appeared many developments providing structures of building models for system communication. Examples include the following systems. The *General AEC Reference Model*, (GARM) a framework for addressing building models issues [Gielingh 88]. The *COMputer Models for the Building INdustry in Europe* (COMBINE) provides a model that confronts building's evaluation of energy and HVAC performance [Amor 95], [Amor 98], [Augenbroe 95]. The *Integrated Data Model* (IDM), the for COMBINE models. The project developed prototypes of the next generation of *Integrated Building Design Systems* (IBDS) based on available product data technology and on the ISO-STEP

standardisation. In the IDM core model, information on building spaces, enclosing elements, and shape description are defined, with special attention to the relationship between spaces and enclosing elements [Eastman 99]. The *RATAS* a national building model by the Technical Research Centre of Finland (VTT). The *RATAS* describes a building symbolically with the use of objects-entities. The object description of a building is made up of objects and a network of relationships between these objects. In this model, no standard method for describing classes such as STEP and particularly the relationships between classes has been used [Björk 92a], [Björk 92b], [Björk 94].

In general the majority of these developments take a great number of criticisms mainly because their approach leads in general to *static* representations [Eastman 94], [Eastman 95], [Galle 95], [van Leeuwen 97]. The *static modelling schema* was focused towards non-incremental product determination. However, the main reason of criticism was that design as a dynamic process it was not possible under these developments. Especially the domain of architectural design requires (given the reasons we describe earlier) information models flexible, dynamic and evolvable capable to confront continuous change of design information. Therefore they are no suitable in a design situation.

The dynamic definition of design object classes was the answer of the above situation. Dynamic modelling schema is a design information model capable to accurately represent the state of design during the design process. Moreover it is capable to contain the semantic of the design paradigm and the meaning of the design decisions [Fridqvist 00]. The building modelling approaches that adopt the dynamic approach are based mainly on a small set of generic object classes. Examples of such developments include the following systems.

The *Engineering Data Model* (EDM). The *Engineering Data Model* was intended to capture the semantics of design and engineering. It was based on an object-relational database, UniSQL [Eastman 95]. EDM is an information model that represents a building in different levels of abstraction and within many building technologies, and satisfies a wide range of building uses. The aim of the *IDEA+* project is the development of an *Integrated Design Environment for Architects*. In thus model design tools and computational tests make use of one and the same core object model [Hendrix 00].

The research project BAS-CAAD aimed to develop principles for information systems architectural design at early stages. The project expands the general concept of building product modelling to include not only buildings and building parts, but also users and user activities, and the relations between buildings and user activities [Ekholm 98], [Fridqvist 00]. The Feature-based Modelling project develops a framework for information modelling that provides extensibility of conceptual schemas and flexibility of instance models. This project deals with information of the early stages of architectural design [van Leeuwen 99].

2.4.3.2. Architectural knowledge in CAD systems

Architectural knowledge could be incorporated implicitly within building models. Such a model would capture and organise architectural elements (space, structure) in relation with specific requirements, rules and constraints. The knowledge that characterises and defines architecture incorporates different building types that have fundamentally different design rules and different performance conditions. Structured and integrated representation of such information it is important mainly among the different phases of the design process, even when only one engineer, (for example an architect), approaches the design of a building during the conceptual phase, or while applying building regulations, or a personal aesthetic way of designing, i.e. a personal architectural style. The integration of ‘*Knowledge-Based*’ and ‘*Computer-Aided*’ design systems provides novel tools for design synthesis and stimulates designers’ creativity. Designers utilise bodies of knowledge and certain operational tools. This fact explains why the developments of Computer Aided Architectural Design (CAAD) systems are so difficult [Simon 96], [Eastman 95]. Knowledge-based supporting CAAD aims at the development of design environments to help in the conceptual stage of the design process [Coyne 90].

A number of approaches are developed. Single building models have been introduced in different CAD systems for the generation and evaluation of new alternative design concepts. The *Software Environment to Support Early Phases in Design* (SEED) project provides computational support for rapid development of design alternatives at the early stages of building design [Flemming 95a], [Flemming 95b], [Rivard 00], [Woodbury 95]. *SEED* is planned to be operated interactively by the architects and engineers involved in the design and it contains mechanisms to enable storage and retrieval of design cases. The *Knowledge-based Assistant for Architectural Design*

(KAAD) was a design aid system. In general, the *KAAD* provides large number of building components with semantic richness, with multiple inheritance, which are open and scalable. The *KAAD* did not determine a single top-down or bottom-up design flow [Galle 95], [Carrara 94]. The *Intelligent Computer-Assisted Design System* (ICADS) project provides a system architecture that enables some Rule-based Expert systems to collaborate in evaluating designs [Pohl 94], [Chun 97]. The *Integrated Building Design Environment* (IBDE) was not a general purpose system. It mainly provides an environment defined by seven independent knowledge-based systems which are specialised for the design of high-rise office buildings [Fenves 94]. Finally, *P3* is a computational environment to support design collaboration [Kalay 98], [Kalay 99]. The *P3* project consists of three computational systems: a shared-product engine, a performance-evaluation system and a process-based component.

2.4.4. Feature-based Modelling

Modelling by Features is an information modelling technique that has been developed initially in the era of mechanical design [Shah 95]. During the early approaches and developments *Feature-based Modelling* (FBM) has been formed by geometry models. However, they used as an attempt to recognise the semantics of design. *Features* are used to generate, analyse, or evaluate designs. Through the use of *features*, may also handle the reasoning of topology and geometry.

Feature-based modelling is used for modelling products. In FBM the basic entity is a feature. A *feature* is defined as a representation of shape aspects of a product. These aspects have two characteristics, they are mappable to a generic shape and they are functionally significant for some product life-cycle phase. However geometric models require information enhancement before they are suited to the exacting requirements of concurrent design. Geometric models alone could not relate higher level characteristics like the function and behaviour of a product. Therefore the essential characteristic of a *feature* is that it has a well-defined meaning for a particular life-cycle activity [Bidarra 99], [Bidarra 00].

The FBM approach offers powerful support for the functional, geometrical and technological description of products. *Features* were mainly employed in engineering design. The definition for *feature* of Shah and Mäntylä is from the engineering point of view: “*Features are generic shapes or characteristics of a product with which engineers can associate certain attributes and knowledge useful for reasoning about*

the product. Feature encapsulate the engineering significance or portions of the product geometry and, as such, are applicable in product design, product definition and reasoning about the product in a variety of applications such as manufacturing planning” [Shah 95]. In general, features model the elementary part of the products. These parts define in turn assemblies which they define complete products. For the representation of a *feature model* is usually utilised a graph and a geometric model of the resulting shape [Shah 95]. In particular, all *feature* instances are contained in the graph with validity constraints about their form and *feature*, attach relations and model validity constraints. The geometric model can be a boundary representation (B-rep), but also a more extended representation from Constructive Solid Geometry (CSG).

2.4.4.1. Features Categories

There appeared many categorisations of *Features* while are almost always application dependent. A main categorisation concerns their function. These categories include *form features*, *material features*, *pattern features*. In general, *features* could be distinguished in two categories considering their degree of complexity. The first category is *elementary Features*, and the second is *compound features* [Bronsvoort 93]. *Elementary features* are basic simple and cannot be decomposed into more simple features. *Compound features* are composed of several other features. Another distinction of *features* is based on their level of detail of the geometric description. The two categories are *explicit* and *implicit features*. *Implicit features* are described by parameters only and they are not evaluated into a precise geometric description, their exact shape is not represented and it is merely implied. *Explicit features* have their shape explicitly described by a geometric model while their resulting geometry is evaluated [Bronsvoort 93]. *Features* have intrinsic and extrinsic properties for their definition. Intrinsic properties concern properties that affect only the *feature* itself. Extrinsic properties affect and depend on the properties of other *features*. Other characteristic properties of the *features* are derived and non-derived properties. Derived properties are such their values do not determined by the user, but derived from other *features*.

Constraints play an important role in *feature modelling*. The management of constraints concerns two approaches distinguished on the basis on how the *features* are represented. When *features* are presented in terms of procedures and rules the

approach is called *procedural design by features*. Therefore the constraints are unidirectional as the change propagation. When the *features* and their constraints are defined in a declarative manner the approach is *declarative design by features*. In this approach definitions are remain more generic and context-independent. The validation of constraints take place after the change has been made to the model [Shah 95].

2.4.4.2. Feature-based modelling approach

The *Feature-based modelling* approaches include three categories of creating a *feature model*, *Feature recognition*, *design by features* and *Feature conversion* [Shah 95].

In *design by features* the designer has the ability to specify new *features models*. The creation of new *feature* instances is possible by specifying and/or modifying values for the parameters of *features*. In *feature recognition* the geometric object of a product enables the recognition of features. The most important categories of feature recognition methods are graph-based, rule-based, volume decomposition and geometric reasoning methods [Shah 95]. The *feature conversion* approach enables the definition of other feature models based on a feature model of a product already created. The new feature corresponds to alternative views of the product. *Feature conversion* is a technique that defines the basis for *multiple-view feature modelling* systems. Such a *feature modelling* direction is a product development approach that combines Concurrent engineering and *feature modelling* [Bronsvoort 01].

In the current *Feature-based modelling* systems exist the four following drawbacks [Bronsvoort 01]. The meaning, or semantics, of features is poorly defined, limiting the capability of capturing design intent in the model. Moreover semantics are often not adequately maintained during modelling. Second, the product development phases lack product models with multiple feature views. In general current only form feature views are supported by multiple-view feature modelling systems. Third, feature modelling systems does not support yet collaboration of several users in developing a product. The last drawback concerns the fact that while products contain free-form shapes, feature modelling systems use mostly regular shaped features.

We refer two approaches from FBM. The first approach has focused towards to overcome the abovementioned shortcomings of Feature models. This approach concerns the development of Semantic Feature modelling [Bidarra 99].

Probably the most important characteristic of the semantic feature modelling approach is that the semantics of all features is effectively maintained throughout model evolution, for all modelling operations.

Semantic feature modelling is a declarative feature modelling approach. This means that, in contrast to many current approaches, feature specification and model maintenance are clearly separated. All properties of features, including their geometric parameters and validity conditions, are declared by means of constraints. The main advantage of declarative modelling is the freedom in the type of constraints that can be specified, and therefore in the way a model can be edited and maintained.

Another characteristic of semantic feature modelling is that the whole modelling process is uniformly carried out in terms of features and their entities (e.g. faces and parameters), and of constraints among these. So all modelling actions performed by the user are effectively *feature-based*, and the same applies to all output, graphical and textual, generated by the modelling system. An advantage of this is that feature's faces and their names are persistent. [Bidarra 00]

The second approach develops a Feature-based Modelling framework for Architecture information modelling. This framework is implemented in the early phase of Architectural design [van Leeuwen 99]. Characteristics of Feature-Based Modelling are very appealing to the dynamic architectural designer who is struggling with ill-defined design problems at the early stages of design. Research on a FBM approach for architectural design has led to the development and prototypical implementations of a theoretical framework with the following characteristics [van Leeuwen 97], [van Leeuwen 99]:

- *Features* are used to represent the semantics of a building design.
- *Features* are the formal definition of characteristics or concepts of design.
- *Features* are applied to multiple levels of abstraction of modelling the design.
- Features can be *Generic Features*, shared by the domain of architectural design, or *Specific Features*, which are defined for a particular view.
- *Types of Features* can be defined by designers as the need to formalise a design concept arises.

- *Features* form interrelated structures in a *feature* model, using the relationships that are defined at the level of *Feature Types*, or by adding occasional relationships at the instance level.

- Libraries of *Feature Types* represent bodies of domain knowledge. In these libraries can also be included instantiated data, mixed with the typological definitions.

The Feature-based Modelling framework is property oriented. The basic modelling objects of FBM represent properties of things in addition to the traditional classes of things [Fridqvist 00], [van Leeuwen 01].

Chapter 3

Architectural Style

3.1. Introduction

In the current chapter main attention is focused towards the exploration of the nature of style and the definition of a style description framework that can help designers to utilise and differentiate styles creatively. This chapter presents an analysis and definition for the concept of architectural style and introduces its principles in the generation of designs during the conceptual phase of Architectural design. The conceptual phase, in a given context, is dynamic as an exploration of emerging relationships between designs, their style and the context itself. We define three stages for the present research:

Presentation of style in different fields,

- Definition of an operational description of architectural style, and
- Introduction of a framework of the description of architectural style and its integration in an information system for the evolution of style and the emergence of new styles.

The first objective is to overcome the difficulty to provide a semantically rich representation of style. A quantification of style is problematic because, information about architectural style is interwoven with information about each of the individual building's elements, its morphology and their internal order - syntax. The second objective is the development of a representation to adequately support the needs of design style development, analysis, and evaluation during conceptual design phase. Such development is based on the separation of the building's style composition from its stylistic components. After a review of different concepts of style, we have adopted a representational scheme. The developed scheme is used in two cases for representing two types of regional – vernacular architectural style, while it captures and organises their architectural entities in relation with their specific principles and constraints.

3.2. The concept of style

Many researchers have approached the concept of style in many areas like art, design, architecture, literature, music. In general, such approaches could be classified in two distinct views, which set the guidelines for approaching the concept of style. Firstly as an *object view*, and secondly as a *process view*.

3.2.1. The object view

The definition of style through an object view depends on the set of common characteristics that products have. The emergence of style is based on a repetitive appearance of a set of ‘features’ in a number of products, such as buildings, paintings, poems, and music. Art historians and critics create classes of styles “*on the assumption that a certain complex set of elements common to a group of works is sufficiently stable, distinct and relevant to justify characterizing it as a style*”, [Chan, 92]. A definition of style in art history through the concept of constant forms is the following: “*in the study of the arts, works are the primary data; in them we must find characteristics that are more or less stable, in the sense that they appear in other products of the same artist(s), era or locale, and flexible, in the sense that they change according to a definable pattern when observed in instances chosen from sufficiently extensive spans of time or of geographical distance. A distinguishable ensemble of such characteristics we call a style*” [Ackerman 63]. According to Ackerman, a list of the characteristics of a work of art includes: first convention of form, secondly material, and thirdly technique. In a similar approach Shapiro, [Shapiro 61] argued that *formal* and *qualitative* characteristics are more decisive for the formation of a style. According to many researchers “*By style is meant the constant form – and sometimes the constant elements, qualities and expression – in the art of an individual or a group*” [Shapiro 61]. For that reason he proposed the following aspects for the description of the style: form elements or motives, form relationships and qualities. The form features are fundamental for the artistic expression. Their existence alone does not elucidate a style. A definite factor for the difference of styles is the alternative ways of combination of such form features. Form features and their relationships not only constitute a coherent whole but also they express the qualitative factors of a style. It is evident, according to the object view, that the study of style interweaves the set of form features-characteristics and their compositional order. An

analogous approach appeared in architectural history by Smithies “*Style may be considered as the collective characteristics of building where structure, unity and expressiveness are combined in an identifiable form related to a particular period or region, sometimes to an individual designer or school of design*” [Smithies 81] and [Pothorn 82].

We can conclude that a constant and recognisable set of features as expressed in forms of manmade objects-artefacts play an important and decisive role for their stylistic categorisation and aesthetic judgment. Form features are a fundamental medium for the expression of the designer-creator’s preferences, aesthetic ideas, et cetera. Therefore, they could be distinguished into different classes in order to define a plurality of styles in automotive design, industrial design, interior design, furnishing design and building design.

Depending on the original appearance and utilisation-reuse of features, the latter characterise an individual style, or the style of a group of designers [Banham 80]. When features appear in a period of time then they define the style of a historic period [Ackerman 63]. If a set of features adopted by designers is recognised in a specific geographic region then it defines a regional style, a vernacular style [Michelis 81]. Considering the object view for a definition of style it is evident that the extraction of any physical form, pattern, or any distinguishable characteristics may not be enough to a complete the description of a style. Although the existence of morphologic features is a fundamental issue for the expression of a style, it has become evident along many researchers that morphologic features alone are not capable to uncover certain intrinsic qualities or principles of architectural styles, whenever they belong in a personal style, in a historic period of time [Buchwald 99], and regional vernacular style [Michelis 81].

3.2.2. The process view

In the process view the formation of a style is achieved through some important factors that characterise a creative process. This approach is concentrated in the way of doing things. During a process three aspects play an important role, choices, constraints, and search orders implemented in a process [Chan 92]. Gombrich considers as an important factor in style the choices during the creative process. He also indicates that: “*The history of taste and fashion is the history of preferences, of various acts of choice between given alternatives*” [Gombrich 60]. Simon argued: “*A*

style is someone's way of doing things chosen from a number of alternative ways", [Simon 75]. For both of them the birth of a style could emerge around a set of choices between ways of performance or procedures. For Gombrich, these choices are of the kind of perceptual cues, while for Simon these choices are those that emerge during the design process. The decisions made among alternative choices characterise a style. The second fundamental factor is the constraints that were imposed during the creative process. Constraints could strongly affect the selection among choices. The third factor that is influential for style is the search order. Simon argues that human designers provide specific procedures for the priority of goal satisfaction and constraints application [Simon 75]. During the design process, the order of satisfaction of certain aspects of design imposes a sequence in the satisfaction of the design aspects. Such a procedure could result in a distinct form and composition for the design artefact.

The derivation of procedural knowledge of style is useful in design. Such knowledge enables designer to uncover the source producing style characteristics of products and to use them for producing designs under that style and/or evolve further a style. Recent studies of the explicit representation of style include the style of Taiwanese traditional house style [Chiou 95], and style of renaissance architect Palladio [Stiny 78]. Both approaches imply the description of style in the form of rules of composition for shape grammars. These approaches can be manipulated by the computer to generate designs with that style.

3.3. Identification of style

There exist three factors that take part in the emergence of style:

- The environment, (cultural, social, technological),
- Design methods and production procedures,
- The impact of imagination on the current human knowledge.

In a fruitful research Chan have made essential achievements in the study of style [Chan 01]. He has concentrated on the style of design products and especially on architectural style. In order to define the forces that generate style, and in particular individual style, he utilises both the object and the process view. Approaching the formation of style from the point of view of common features and the design process, he argued that a style results from executing fixed sequences of design goals (design

method), applying fixed sets of constraints (design knowledge) at each goal stage, and exercising preferred presolution models and primitive forms. In one of his studies [Chan 94] he had limited the concept of feature to cover only morphological aspects of design products. In more detail, he claims that a feature, in order to be a stylistic feature, it has to comply with the following properties:

- It is a form or composition distinguished by some particular configuration, or proportion;
- It has some contextual relationship with other features;
- It is originally generated by a designer through certain creative processes;
- It is adopted or copied by a designer from other sources;
- It is a member of a set of prominent forms repeatedly used by a designer.

In another study about the formation of architectural style he argues that style could be measured [Chan 00]. For that reason he has employed as a fundamental unit for categorising a style, a set of common features appearing in design products. Therefore, a style could be identified in such a way. Next, he exploits the common features approach defining a measurement of the strength of a style, and the degree of similarity between two styles. All objects that possess the same set of features should belong to the same style category.

3.3.1. Measurement of style

The number of common features is fundamental for the recognition of a style. The measurement of style is defined when a minimum number of features appears. It is possible to have buildings belonging to the same style with more features than others [Chan 00]. Results of psychological experiments, [Chan 94], exploring the definition of style conclude when three features appear in products then a set of common features could be defined. Chan concludes that *“a style is no more recognisable when the number of features is three or fewer, despite the contents of the features and is measurable when there are more than three.”* [Chan 94]. Furthermore, a significant parameter for style alteration is the topological distortions of the syntax among features. When topological relations are changed then the characteristics of a building are changed. Therefore, a feature will no longer represent a style [Chan 94], [Chan

00]. On the other hand the geometrical distortion of a feature could be up to 40% and still the feature is being representative for a style.

3.3.2. Degree of style

The definition of the degree of a style is relative to the number of the apparent features existing in an building. Degree of style could be defined in two ways, within the class of a style and between classes of styles, [Ding 01]. The degree of style is strongly influenced by the both quantity and quality of the repetitive features that appear in a series of buildings. This set of common features determines a pattern recognition that could characterise a specific style. In general, the greatest the quantity, the more easily determined is the image of a style. Two factors that determine the degree of perception of the common features are the following, (1) the size of the features in an object, and (2) the significance of perceptibility, [Chan 00]. The strength of a style could be represented by a number of common features. Therefore, different strength between two styles signifies the degree of each style. The second factor of the degree of style is the quality of these common features. It is obvious, then that large-sized and characteristic features are more easily recognized for a style.

3.3.3. Style in Architectural design

The impact of style during architectural design is very significant. Buildings are artefacts whose aesthetics is a major evaluation criterion. Building is the final determinant of aesthetic judgment, where aesthetic considerations like harmony, balance, unity, of form(s) have a strong impact in this judgement. In general, style could aid towards the reduction of alternatives in the problem domain [Akin 86]. A style could be conceived as a system of constraints by architects. In this way, a system of constraints that refer to a particular style limits the decision within a smaller part of the problem space [Chan 94]. Another significant impact of the utilisation of style is the fact that it is used by designers for the reduction of the solution space. As presented in Chapter 2, design problems generally do not have unique optimal solutions. Therefore style is used for the selection of one among many satisfactory solutions. In this study we will concentrate stylistic constraints in general as soft constraints. The utilisation of styles suggests design constraints that direct the way choices are made during the design synthesis. Another role of style is the implication

of design principles during the spatial and morphology design composition. Styles express a particular mixture of various design principles. The quantities of such principles define alternative styles while a small difference on such a quantity results in a different style. We concentrate on regional – vernacular styles although the factors that determine the formation of a vernacular style are not that clear as the above factors that determine the formation of individual style.

3.3.4. Architectural style

Style is very influential during the design process of a building. We concentrate our interest in the architectural design of buildings. Architecture is highly interpretive. Architectural style defines the structure and expresses the character of any space [Krier 88]. The stylistic formation of a space – building is one of the most complex and intrinsic of the architectural design problems. Style strongly characterises a building as it enables the designer to transform it by grouping or classifying existing design-building elements according to some distinguishable properties and patterns [Michelis 01]. The context or the situation is influential in transformation processes that may occur incrementally through acts of imitation or radically through acts of innovation.

In the design process of a building the architect introduces and defines certain constraints for the expression of a variety of meanings and the satisfaction of design constraints. The repetition of common morphological features defines a character for the artefact, in this case a building, while the diverse set of features could enable the appearance of different architectural styles. Therefore, principally an architectural style is defined as a set of common features appearing in building(s). A building contains a set of distinguished features. These features could represent the particular style of that building. Therefore, a building could be decomposed in structural and spatial elements. The global morphology and the local morphology of the building elements and the relationships between both structural and spatial elements, evoke the specific spatial qualities of each architectural style. These stylistic attributes are functions of a design process. A style is a function of the design process that generates these features and their relations. Chan express a similar position with that of Simon [Simon 75] which argue that during the design process dynamic forces appear that generate style [Chan 92]. In that sense architectural styles are not simply some catalogues of patterns or concepts that are applied to a design situation. Rather the

process is one in which these concepts are as much shaped by the situation as the situation is shaped by the concepts. Morphology, like function, can also be regarded as a very general type of information, relevant to almost all elements in a building design, and even to many non-physical concepts [Makris 01]. Different aspects of morphology can be distinguished, which leads to a decomposition of its concept type into more detailed concept types for ‘*shape*’, ‘*position*’, ‘*orientation*’, et cetera. An architectural style provides knowledge of:

- Types of stylistic features,
- Organization principles, which define on a high level of abstraction what principles to employ in the actual placement of spatial and structural elements of a building.
- Morphological principles, which define the form of a building through establishing a general outward shape.

Therefore the role of style in architectural design is twofold; firstly it works as a synthesis principle allowing the emergence of building designs while it provides imaginative order within a complex-non-measurable but ordered design domain. In the history of architectural design, with the aid of style it is possible to describe consistencies among building projects as creations of an individual architect, school, cultural period or geographic region. Studies on consistencies or patterns of buildings and spatial features are a common approach to investigate and model architectural analysis and criticism [Michelis 01], [Clark 96]. In the following analysis, architectural style can be broken down in features and principles. A fundamental issue in any architectural concept of building, and in particular the style of a building, is first the representation of its solid constructed region as well as the spaces that are enclosed within that constructed frame [Miess 90]. A second fundamental issue is the kind of relationships that are applied between these spaces and structure, as well as the relationships among the spaces themselves [de Ven 87].

3.3.5. Morphological features

Morphological features that define a style are repetitive natural parts in a group of buildings. These express the physical properties of the building’s space. That repetition categorises and characterises the specific group. Further studies showed that among features these concern *syntax* and *topological* relationships, both are crucial

factors for sustaining a style. Any possible changes of syntax would strongly influence style recognition. The unique composition of these elements can define a style. A set of morphological features includes a number of sub-features and relationships between them. In architecture, morphological elements are in general architectural 'dictionaries'. Typically, architectural features include columns, beams, arcs, walls, roofs and spaces. These are functional features and could be further analysed in two categories as:

- *structural* elements (columns, beams, arcs, walls, roofs et cetera);
- *Compositional* elements (spaces, rooms).

As an example we can consider, the hemispherical dome that appeared in most Byzantine architecture. This recurrent feature could conclude to the assumption that a church belongs to the Byzantine style in architecture if it has the hemispherical dome under certain circumstances [Michelis 01], [Buchwald 99]. Apart from functional elements there are some other features that define a *visual presentation* of a building. These are lines, surfaces, planes and volumes. For instance the volume penetration is important attribute in churches of Byzantine style (Figure 3-1).

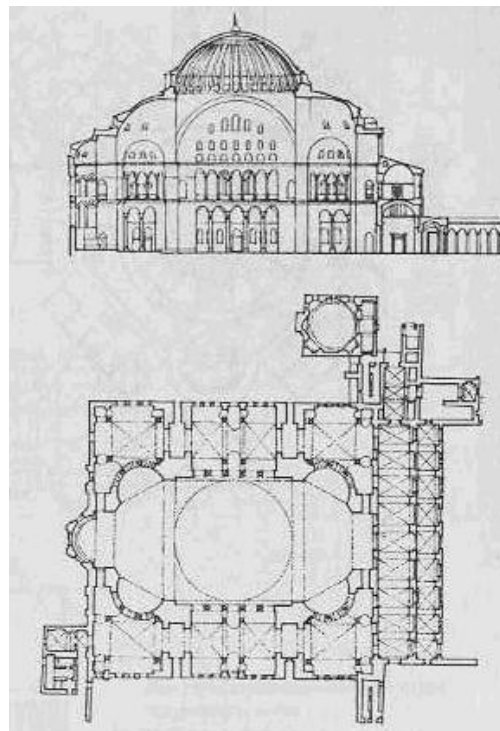


Figure 3-1 Byzantine style

Of equal importance for style classification are some properties that a building could also be characterised by. Such a group of *properties* like colour, magnitude, materials, texture is highly definitive of a style. As an example, we can consider the ancient Greek style representing the classical order. According to this, the features of the Greek temple could be decomposed into parts like: structural (columns, stylobate), morphological patterns (tympanum, metope), and material (marble), (Figure 3-2).

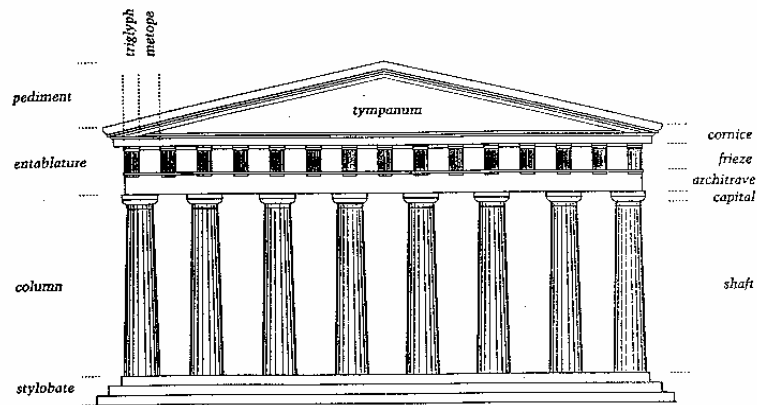
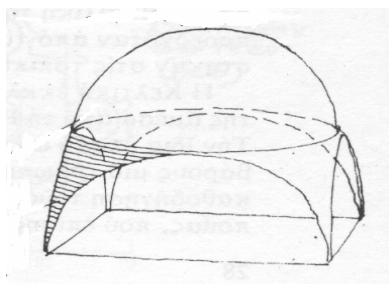
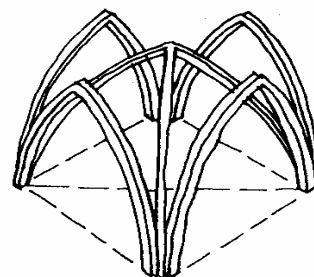


Figure 3-2 Greek temple

In the same manner emphasis is given not only to elements but also especially to systems of forms. The repetitious features define artefacts, which give existence to the forms of a style. Such elements could be recognised as the round arch of Roman and Byzantine church style, the pointed arch of Gothic architecture, (Figure 3-3).



Round arch - Roman and Byzantine style



Pointed arch - Gothic style

Figure 3-3 Byzantine – Gothic style

There exist two groups of features, simple and complex. In general a simple feature refers to a simple geometry, while a complex one refers to a combination of several

simple features. The concept of stylistic feature could be enriched by taking into account not only morphological features and their relationships that exist between them, but also any compositional method-strategy for building design. In our analysis we will suggest that each style could be characterised by a distinct set of form – relationships features.

3.3.6. Semantic features

The synthesis of the *object* and *process view* for style definition leads to *style semantics*. Style semantics are expressed through semantic features. They represent those meaningful qualities that also characterise a style. A group of researchers, [Ding 01], approach the interpretation of architectural style using *syntax – semantic* model. They consider architectural style as a representation of some meanings within a group of designs. According to them, it is possible to regard a style as a representation of common particular meanings called *complex semantics* expressed from a set of designs. Style could be analysed into an ensemble of complex semantics, (Figure 3-4). The latter emerged by the synthesis of some simple semantics after the articulation of a group of forms. The choices of the specific forms, (or perhaps the creation of some new forms), are directed by the design decisions. The notion of simple semantics could derive from *properties* of forms (geometric) and *relationships* (topological, organizational) among them.

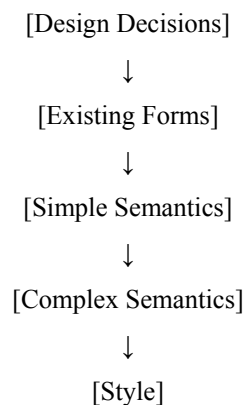


Figure 3-4 The generation of style from design decisions

Complex semantics are regarded as principles higher ordered, (like axially, horizontality and mirror symmetry). The origin of these principles is a set of decisive

formal patterns that could be observed in a building or in a group of designs. On a lower level of organization these morphological – spatial patterns may derive from associative appearance of structural or morphological elements in groups. The content of complex semantics could suggest psychological, interpretative and aesthetic aspects that characterize the whole presence of the design(s). For instance Byzantine style refers to a set of common complex semantics: dynamic line, emphasis on dome, openings in the base of the dome for visual and structural lightness, structural framework using stone to concentrate weights and stresses, [Buchwald 99]. Design decisions introduced describing particular lengths, widths, parallelarity or verticality, the application of some building codes, the composition of a geometric layout, et cetera. For the purpose of the present research, we present a characteristic example of a vernacular architectural style, that of Metsovo houses. The spatial (compositional) and morphological languages define a set of characteristic complex semantics: dynamic configuration, symmetrical volumes, predominance of space over form, et cetera (Figure 3-5). These complex semantics are derivable from a set of simple semantics as: roof following the volume of the house, horizontal axis, a-symmetrical and natural materials.



Figure 3-5 Metsovo house style

The buildings are composed by specific simple architectural forms, which in turn express some simple semantics. The builders made some common decisions relating to form elements and spatial relationships between form elements. In particular,

- ‘*Union of roof and the volume of the house*’ (manipulation of roofs element). A result from the decisions concerns particular analogies in lengths, widths and parallelarity.
- ‘*Horizontal axis*’ (penetration of forms) results from the geometrical plan with the emphasis given on two axes.
- ‘*Symmetrical*’ (layout of spaces) results from decisions concerning the spatial planning of each space.
- ‘*Natural materials*’ (derived from the forms) results from the choice of materials and from the revelation of the nature of these materials.

In a broader regional area similar design decisions were made in many of other house designs. During the design processes, similar manipulation of forms were included resulting, however, in different compositions of spaces. Nevertheless, the result was a specific style recognized with the name Metsovo.

3.4. Modelling architectural style

In order to model style we first need to propose a working definition of it. Architectural style defines the structure and expresses the character of any building, [Michelis 02]. The stylistic formation of a building is one of the most complex and intrinsic of the architectural design problems, [Chan 01]. For the aims of this part we will introduce two specific regional architectural styles Santorini and Metsovo [Michelis 81]. Architectural styles are not simply some catalogues of patterns or concepts that are applied to a design situation. On the contrary, this process is one in which these concepts are as much shaped by the situation as the situation is shaped by the concepts. Morphology, like function, can also be regarded as a very general type of information, relevant to almost all elements in a building design, and even to many non-physical concepts. Architectural styles are defined, essentially, by their:

- Constitutive distinguished elements,
- Relative position-placement, and
- Morphology.

Every style requires a set of characteristic features and principles of their relative composition. Therefore an architectural style provides knowledge of:

- *Organization principles*, which define on a high level of abstraction what rules apply in the actual placement of spatial and structural elements of a building.
- *Morphological principles*, which define the form of a building through establishing a general outward shape.

A fundamental issue in any architectural concept of building, and in particular the style of a building, is first the representation of its solid constructed region, the spaces that are enclosed within that constructed frame, [Joedicke 85]. Second, of same significance is the kind of relationships between spaces and structural elements, and the relationships among the spaces itself, [Joedicke 85].

Based on the discussion presented in the current and previous sections, we accept that architectural style is composed by:

- A set of *features* represented by a set of spatial and structural elements
- A set of *principles* which are represented by constraints in the form of properties of and relations among the aforementioned elements.

This decomposition will be adopted in the following analysis. For example, a set of structural and spatial elements (e.g. roofs or rooms respectively), could introduce, when combined, like symmetry, rhythm, et cetera. The latter are represented by the corresponding *principles* in the current model whereas, on the other hand, each one of these elements, when considered separately, is represented by a *feature* in the current model.

3.4.1. Stylistic Features

We define two categories of stylistic features representing the two categories of building elements: spatial and structural. The first category concerns elements like roofs, walls, et cetera. The second category concerns elements like rooms, open spaces, patios, et cetera.

3.4.1.1. Spatial Features

The category of spatial features includes physical or virtual spaces that satisfy a function according to the program (brief) of a building. Examples for such functions define spaces for sleeping (bedrooms), living (living rooms).

3.4.1.2. Structural Features

Structure is intentionally used in many formative ways to reinforce the emergence of a particular style. The category of structural features includes all objects for the definition of the structural design of a building. Examples of such features are those that concern the structural design of a building, bounding elements, such as slabs, (ceilings, floors), walls, columns, and beams. Such structural form patterns define both the properties and the character of the space enclosed. A frame of structural features could be decomposed into simpler structural elements.

3.4.2. Stylistic Principles

A style provides relationships for the placement of its elements, and it obeys their internal relationships. Regarding architectural style, these relationships specifically determine both morphological and structural-functional properties, (e.g. such as the precise place and internal proportions of roofs for a style of a house) of a building. The principles of an architectural style comprise the following kinds of relationships: those between spatial and structural elements, between spatial elements, and between structural elements. These three kinds of relationships can be classified in two categories: the first category determines topological principles whereas the second category determines formative conditions. As spaces are decomposed by other spaces, they introduce their organisation concerning their activities. The architect, in order to define a functional efficient spatial arrangement, could use ergonomic studies. But what characterizes any style is that while ergonomic and functional rules are satisfied, the personal choices for a characteristic spatial or structural, (both two-dimensional, and three-dimensional), layout define their personal architectural style. Topological relations could also assign relative value to different spaces. They imply relational conditions like public – private, served – servant, individual – group. Topological objectives define the location of individual stylistic elements, affecting how one space or structural component relates to each other.

3.4.2.1. Topological Principles

Topology refers to logical relationships between stylistic features. Relational representation assists the architect in the definition and synthesis of elements. The relationships between spaces are very important for the formation and representation of an architectural style. The spatial organizations are expressed under the concept of

topological relationships. Any topological identity could be emerged with regard to different functions – activities that each group of stylistic elements may satisfy. Any topological scheme could then be decomposed into single areas. In this way it is possible to form a hierarchical order of the interior spaces. The topological category determines the relationship(s) between individual stylistic elements, (spaces and building components) mainly without regard to the dimensions of any building element. Topology defines constraints for the geometric design space. For example, a topological constraint that “*space x is adjacent to the north wall of space y*” restricts the geometric coordinates of ‘*space x*’ relative to ‘*space y*’.

Topology relations include:

- *Adjacency*. It concerns spatial continuity that occurs between two (or more), adjacent elements. Elements can share a common edge and can pivot about that edge. Elements can have corresponding planar surfaces, which are parallel to each other.
- *Orientation*. It determines the direction of an element, relative to ground plane and compass points.
- *Proximity*. It concerns the distance and closeness between two or more elements.
- d. *Interlocking*. This relationship concerns the interpenetration of elements.
- *Schematic axial system*. A schematic axial system introduces lines around or along which, elements are arranged. It defines a spatial system of the building. In general, it indicates ordering principle -principles- for the stylistic elements. An axial system concerns the way of planning the synthesis of a building. It could define orientation and relative position within the layout of the building. A combination of axes could define the development of a specific 3D layout.

3.4.2.2. Formative Principles

This category introduces the formative constraints of a style. Formative conditions introduce manipulations of the geometry representations. These constraints affect both the spatial and formal level of a design. They specify and establish the form language, frequency, complexity, and variation of the characteristic forms of a style. The formative category is also presenting the physical dimensions of the building elements. It expresses not only the dimensions of elements, but it also identifies characteristics concerning area, proportions, et cetera. The form of an element, (e.g. column or space), could determine the overall stylistic organisation of a building.

Formative constraints introduce manipulations of the geometry representations. Manipulations like rotation, shift and overlap, enable the appearance of many alternative configurations. The definition of a style is also based on geometric transformations, such as multiplication, combination, subdivision, manipulation, and derivation. Formative category includes:

- *Symmetry*. The basic characteristic is that the same feature occurs on both sides of the line of symmetry. It implies an axis or a centre about which it is developed. In particular, it defines an arrangement of equivalent features (or patterns of features), on opposite side of a line, about a centre, or an axis.
- *Rhythm*. It refers to any movement characterized by a pattern recurrence of elements at regular or irregular intervals. Rhythm suggests the fundamental notion of repetition as a compositional device to organise elements, (forms, spaces, structure), in an architectural style.
- *Repetition*. Group elements composed, in architectural styles, following a repetition pattern according to their closeness or proximity to one another. Linear pattern of redundant elements is a simple form of repetition.
- *Progressions*. This type includes the following sub-principles. *Hierarchy*. The rank ordering of elements relative to the range of an attribute, such that importance or value is ascribed according to the presence or absence of the attribute. *Reduction*. The scaling down of element(s), in relation with an original or primary element. *Transition*. The incremental change of a property (e.g. length) within a finite limit.
- *Configurations Patterns*. They define the relative disposition of groups of elements in patterns like: central, linear, cluster, concentric, nested, double centred, binuclear. The elements that constitute these configurations need not to be similar. In architectural style these configurations play the role of themes that provide the potential for composing space and organizing groups of spatial, formal, and building elements.
- *Proportion* system. It defines a field with specific points, proportions between lines, and particular angles, according to a specific architectural style. It defines any proportion between the constitutional parts and/or the position of elements of the building. It provides a geometric structure which governs relations and positions. As

stylistic elements combine in a design, the proportion system determines the relative dimensions of the spaces and /or roof, wall along with their placement.

3.5. Two case studies of architectural style

In our proposal for a declarative CAAD system we incorporate all relevant design aspects as functional, spatial, stylistic dependencies during the stage of conceptual design. In particular the proposed model provides distinct views of representation describing topological-spatial, geometrical, and structural aspects of architectural style(s). Following that approach, there is an advantage that different stylistic aspects of a design can be easily distinguishable in the model.

The proposal of the following features and principles follows our earlier analysis concerning architectural style. In particular, in order to regard any legitimate element as stylistic, considered as a feature and also participating in the corresponding principles, for each of the following paradigms, we employ Chan's [Chan 00], working definition about properties of features:

- It has a form or composition distinguished by some particular configuration and a contextual relationship with other features.
- It is a member of a set of prominent forms repeatedly used by the designer.

For the needs of the current research two paradigms of regional traditional architectural style will be modelled. The first paradigm concerns the style of Santorini. In the Greek island of *Santorini* houses are built following a specific architectural style [Michelis 81]. This kind of building is chosen because it satisfies specific topological and formative constraints that can be interpreted as stylistic principles. The second paradigm concerns the style of *Metsovo*. In the west midland of Greece, (village of Metsovo), houses are built following restrictions of the mountain environment again suggesting a set of stylistic principles. Both styles introduce a specific typology of spaces, and their shape is expressed through characteristic geometrical primitive forms.

3.5.1.1. Style "Santorini"

Most of the Santorini edifices share a relatively simple primary form. In the following we consider the building as a volumetric composition. Nearly all of the existing habitations belonging to this style can be broken down to the following elementary,

volumetric set-up: a rectangular volume, topped by a vaulted roof. Santorini, houses have a specific composition of spaces, while their shape is expressed through characteristic geometrical primitive forms, that of a quadrilateral (cuboid) (Figure 3-6). In particular, they have a specific typology of spaces. The composition of spaces is either a rectangular plan, or deeper than longer plan. On this elementary level, variation is manifest in the sizes and proportions of these coupled volumes (width, length and height). Some buildings are modest – consisting of a relatively small ground floor, topped directly by a roof. In general the width of the spaces is greater than its length while the spaces have the same length in many cases. Another basic characteristic is that the spaces do not overlap and they are adjacent. A typical house, usually, consists of two buildings: (i) the main building containing a living room, a bedroom and the kitchen and (ii) the bathroom which is placed outside of the main building. In many cases buildings have a separate ground floor and first floor. The roofs play a prominent role in the perception of the composition as a whole, whereby different roof forms are applied. Among the special characteristics of Santorini's architecture is that every room has either vaulted or flat roof only (Table 3-1).

| <i>Topological</i> | <i>Formative</i> | <i>Formative</i> | <i>Building</i> |
|---|---|---|-----------------------------------|
| Bedroom and living room are <i>adjacent</i> . | Bedroom and living room are <i>coaxial</i> | Building plan is wider (deeper) than long | Santorini house has two buildings |
| Kitchen is <i>placed on the east wall</i> of living room. | Bedroom width is <i>equal</i> Living room width | All rooms are deeper than long | Building is single floor |
| Toilet-unit is <i>placed on the east side</i> of living room. | All rooms has <i>same</i> height | All room have <i>quadrilateral</i> form | Building_1 has living room |
| Bedroom is <i>placed on the north wall</i> of living room. | Bedroom length is ω times Living room length | Room roof has <i>vaulted</i> form | Building_1 has bedroom |
| Rooms are <i>coplanar</i> | Kitchen width is μ times Living room width | Room roof has <i>flat</i> form | Building_1 has kitchen |
| | Kitchen width is ν times Living room width | | Building_2 has bathroom |

Table 3-1 Santorini style

A building of Santorini style usually combines a number of vaulted and flat roofs. The main characteristic of the roof morphology is that the roofs extend towards the longer dimension of the building. In the front part of the building appeared twin long roofs that follow the length of the building, while a flat roof in general covers a space in the

rear side of the building. Sometimes it is possible for more than one space to have a flat roof.

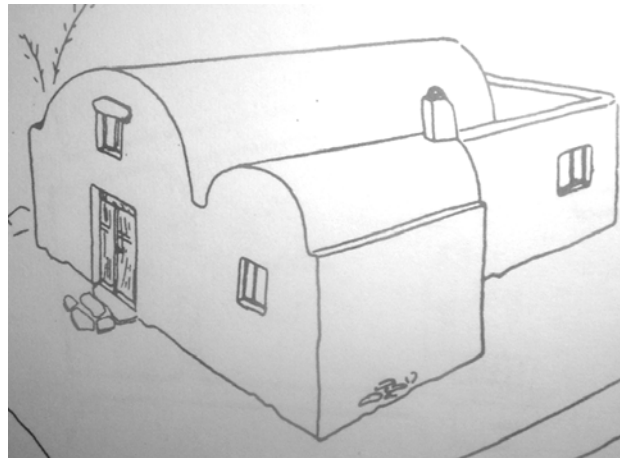


Figure 3-6 Santorini habitation

3.5.1.2. Style “Metsovo”

We consider the building as a basic volumetric composition, the building as an object. Metsovo style houses are built following restrictions of the mountain environment. Most of them share a relatively simple primary form. Nearly all of the existing habitations belonging to this style can be broken down to the following elementary, volumetric set-up.

The regional architectural style introduces a specific typology of spaces, and their shape is expressed through characteristic geometrical primitive form, that of a quadrilateral (Figure 3-7). On this elementary level, variation is manifest in the sizes and proportions of these coupled volumes – width, length and height. The Metsovo style has a distinct character that it is longer than wide. Some buildings are modest – consisting of a relatively small ground floor, topped directly by a roof. In general the length of the buildings is greater than its width. Among the special characteristics of this architectural style is that the upper floor is covered by gable roof while the reception area’s roof is placed vertically to the longer dimension of the building. Another important particularity of this kind of building is that it offers a paradigm of space overlapping: The kitchen and the bedroom in many cases share the same space area. The plans of the habitations are in general of rectangular shape.

The most significant characteristic of the plan is the fact that the part of a room or two (in general the living and /or the guestroom) is overhang for better view.

A typical house consists of two buildings: (i) the main two-floor building that satisfies the principal habitation functions: reception area, living room, bedroom and kitchen (upper floor) and storage place with cellar (basement), and (ii) the bathroom which is placed outside of the main building.

The roofs play a prominent role in the perception of the composition as a whole. Except habitations different types of edifices share the same style principles (Table 3-2). A building of Metsovo style usually combines two roofs. The morphology concerns types of cable roofs. A long roof that follows the length of the building, while the second in general covers a central space and it is placed in a vertical position relative to the main roof.

| <i>Topological</i> | <i>Formative</i> | <i>Formative</i> | <i>Building</i> |
|---|---|--|---------------------------------|
| Bedroom and living room are <i>adjacent</i> | Bedroom and living room are <i>coaxial</i> | Building plan is longer than wide (deep) | Metsovo house has two buildings |
| Kitchen is <i>placed on the east wall</i> of living room | Bedroom width is <i>equal</i> Living room width | Building plan is rectangular | Building is single floor |
| Toilet is <i>placed on the north side</i> of living room | All rooms has <i>same</i> height | Rooms are deeper than long | Building_1 has living room |
| Bedroom is <i>placed on the north wall</i> of living room | Bedroom length is ω times Living room length | Room have <i>quadrilateral</i> form | Building_1 has bedroom |
| All rooms are <i>coplanar</i> | Kitchen width is μ times Living room width | Room roof has <i>gable</i> form | Building_1 has kitchen |
| | Kitchen width is ν times Living room width | | Building_2 has bathroom |

Table 3-2 Metsovo Style



Figure 3-7 Metsovo habitation

3.6. Computer-supported approaches for aesthetic design

In this section we discuss some relevant effort and approaches on design by style. These efforts involve a number of computer approaches for style/aesthetic design. The following research efforts confront with the problem of associations between aesthetic character and shape parameters.

The understanding of design style will be a great advantage in developing advanced *Computer Aided Conceptual Design – Computer Aided Design (CACD – CAD)* tools in design [Sequin 05].

Aesthetic and in particular stylistic concepts are very important in every design product. Many researchers underline the significant missing of systemisation and formalisation of aesthetic related design knowledge [Breemen 98]. The current technology as reflected in commercial *CACD/CAD* systems is concentrated around fast shape prototyping while the visual intent is expressed by manipulation of shape. During the phases of product design process a design style is almost always strongly linked with shape geometry. The latter defines in turn the spatial morphology of the product. In general, the styling tools in product design are limited to tools for geometric manipulations. Geometric modelling engines are based on feature technology. In a research team from Delft University [Breemen 98] has provided examples of possible associations. However the main problem it was the fact that same aesthetic properties can be associated to different shape parameters. As a result

it remains difficult to provide an absolute definition of aesthetic character, which makes very inconvenient the explicit linking with aesthetics knowledge.

In an extended research program academic researchers with industrial partners approach the styling process in product design. Styling is a creative activity where the designer's goal is to define a product that evokes a certain emotion while satisfying obligatory ergonomics and engineering constraints. The main objective of the program was the improvement of the working procedures and the computer aided tools adopted from designers for modelling product shapes. The new modelling tools help Computer Aided Styling/Computer Aided Design (*CAS/CAD*) operators (forming tools called *surfacers*) to easier attain a model with specific emotional characteristics according to the stylist's intent and to preserve them during engineering optimisations. This approach enables the development of tools to preserve the aesthetic design intent during the required model modifications [Podehl 02]. Furthermore with the aid of such tools the aesthetic character from *CAD* models it could be extracted and compare it to others and/or directly act on it. Their work it was based on the analysis of the design activities carried out with stylists and Computer Aided Styling operators (*surfacers*) both in the automotive field (BMW, FORMTECH, PININFARINA, SAAB) and in the field of household appliances (ALESSI, EIGER, FORMTECH). In general the European project FIORES-II (GRD1-1999-10785-Character Preservation and Modelling in Aesthetic and Engineering Design) (FIORES-II), aims at building innovative *CAD* tools more adhering to the creative user mentality and at improving the cooperation between the main actors involved in the product development process, by identifying the relationship between shape geometry and aesthetic character. In this way the aesthetic character will be available in a multi criteria approach for styling and engineering design optimisation [Stahl 01]. Software is developed and is testing according two directions. First, to provide end-users with aesthetic features manipulators for improved and faster achieving the desired changes in the geometric model, conforming to their intent. Secondly, to enable for a deeper comprehension of the geometric characteristics influencing the perception of shapes and of their similarities from an aesthetic point of view [Giannini 03].

The approach of Chen and Owen concerns the development of a formal model to communicate stylistic concepts to the computer for form generation, [Chen 97]. In that research work designers are provided with a language that can communicate style

to computer. Secondly a framework that aids designers in analysing the stylistic attributes of design products, while designers accumulate knowledge related to style. Firstly the '*Style Profile*' is a mechanism for comprehensive formal style analysis. Secondly, this mechanism could records essential properties of styles, and it could serves as a framework for style knowledge accumulation. Thirdly, it is a data storing structure and a data communicating language. Such infrastructure enables the communication of stylistic concepts to computer for form generation of products.

In this work in order to describe styles they use concepts of semantic differential and class-subclass relationships under the name *Style Description Framework*. In particular for the description of style associated attributes they use Style Profiles with polar adjective pairs. For the refinement of the scope of a style they use Confidence factors. Finally, for distinguish weights of the attributes the use Importance Indices. In this approach styles have the characteristics of objects (in computer science terminology), and can be retrieved, modified and instantiated with ease by computer.

In a series of efforts researchers argue about the value of feature geometry for design style [Wang 03]. Their research is focused on the impact of form features on style through the topological structure and geometrical variation. They discover that features are very influential on style formation. Furthermore this result validates their theoretical basis that is based on the earlier research on style by Chan. In particular, they evaluate the influence of design style by feature geometry. In their observations they argue that the geometric manipulation of profiles and paths enable both the creation and transformation of a style. Additionally compositional ordering expressed through topological structure of features affect definitely the design style. Therefore a combination of topological and geometrical attributes on both levels of global and detail perception could enable a definition of a design style. Their remarks are based on an empirical study on two types of typical products chosen with perceptual style. In this way it is possible that a specific style could be achieved. Furthermore it could be easily applied into a new product model in order to be adapted to that style [Wang 02]. Moreover that approach did not imply style in a generative system. The user applies a set of procedures on stylistic features of an object. In general they deal with objects like swords, and statues heads.

3.7. Discussion

The impact of style is very influential during the architectural design process. It can be a personal, historic, regional, vernacular or other style that can define many essential guidelines for the design of a building. Style aids the designer to make many significant decisions especially during the early phase of design. In fact the actual design of a building is strongly inspired by stylistic principles. The introduction of style within a computer aided generative design system could be beneficial during the conceptual phase of building design. As it is drawn from the literature stylistic principles could be characterised in general as *soft* constraints. We provide an operational definition of architectural style and then we define a representation framework for it. Adapting an approach of merging the two views of style, the object and process view, we finally define style as set of features and principles. As features we consider the spatial and structural elements of a building. As principles we consider specific relations among these features, and their properties. Our aim is to express these features and principles in a computational form that can act as evaluation criteria. This form can subsequently be used for the adaptation of a building into a particular style during the conceptual design process. Furthermore, this form could be interwoven within a multi-criteria optimisation computational generative method.

Chapter 4

Research Statement

4.1. Introduction

This chapter is organised in three sections. In the first section we will analyse and develop a framework for the introduction of Architectural Design knowledge in a declarative design system. In order to develop better *Computer-Aided Architectural Design (CAAD)* environments it is important to model architectural design knowledge. Architectural design knowledge is a particularly challenging information domain to represent. In particular, we combine the representation of information as it is used in declarative scene modelling and architectural conceptual design. A declarative process could capture the nature of the early phase of architectural design. During that phase, architects use visual and verbal descriptions in order to express their ideas. We develop a framework for *Declarative Knowledge for Architecture-oriented Building Modelling (DKABM)* to incorporate knowledge from Architectural design to the conceptual design of scenes.

In the second section the proposed architectural style scheme from Chapter 3 is introduced within the declarative design system, with the aid of knowledge modelling framework. It is also analysed how it is possible to affect the three phases of the declarative conceptual cycle. The architectural knowledge includes stylistic information for the creation of buildings. Different styles of each design or from different designs could be stored in a *Style Library* and used for the advantage of future design cases. The adopted notions of architectural style in the proposed structure will further facilitate its incorporation in an evolutionary declarative design system.

In the third section we present the multi-objective genetic algorithm (MOGA). We focus on the development of a new generation engine that is based on a multi-objective optimisation genetic algorithm within MultiCAD. The evolutionary declarative design environment will direct the development of new designs in two interconnected cycles. The first cycle will concern the evolution of the spatial

planning of a building design; the second cycle will concern the evolution of roof morphology of a building design. In parallel the new generation engine will provide with two discrete development processes. The first process will concern specific tasks for the developer of the system. It will enable the developer to define architectural styles as objective criteria. At the same time it will provide the developer with a method for the extraction-definition of the degree of importance for such stylistic objective criteria. Finally it will enable fine tuning of the parameters of the *MOGA*. The second process will concern a specific design environment for the user of the system. The user will provide declarative descriptions and with the use of the *MOGA* will result in design solutions adapted to a particular architectural style. A model of an evolutionary declarative design system is analysed in a specific design context of implementation, that of architectural conceptual design.

4.2. Architectural knowledge in MultiCAD

The motivation of this work was to take advantage of the benefits offered by declarative modelling and combine them with knowledge drawn from the domain of Architecture. In specific, the intention has been focused on:

- The proposition of an appropriate framework for building modelling comprising architectural knowledge, and
- The introduction of the knowledge drawn from a domain, such as Architecture, in an information system that is based on declarative modelling.

For the aims of this research, we introduce a specific architectural paradigm, which concerns a building for habitation. It will be introduced in MultiCAD information system proposing and depicting a new way of development for the knowledge domain-specific approach of such systems. MultiCAD is a software architecture framework for the development of intelligent information systems that support the declarative design process. MultiCAD is a multi-layered architecture. These layers are projected to the three phases of the declarative conception cycle: Description phase, Generation phase, and Scene understanding phase.

The latest version of the conceptual model of MultiCAD architecture is based on the *Extended Entity-Relationship (EE-R)* model: It extends the Entity-Relationship model theory in order to include notions as aggregation, inheritance et cetera. The scene's description is stored in the relational-object database as an assembly of *objects* having

properties and being related to each other through *relations* (*Scene Meta-model* shown at Figure 4-1).

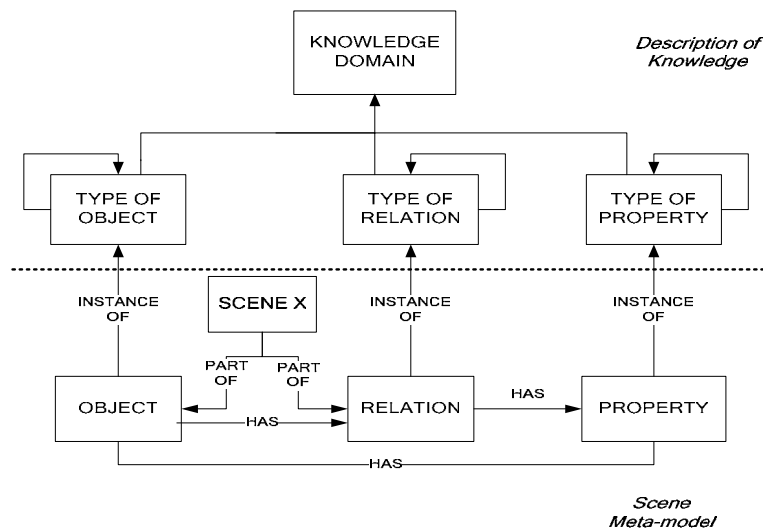


Figure 4-1 Scene Meta-model and Knowledge description in MultiCAD

Furthermore, the Knowledge Base contains a series of types of objects, types of properties and types of relations constituting the description of Knowledge in MultiCAD (Figure 4-1). The elements of the Scene meta-model are instantiated in the Knowledge Base.

4.2.1. Architectural knowledge framework for building modelling

The Declarative knowledge framework for Architecture-oriented building modelling (*DKABM*) is proposed and discussed. The framework shares some characteristics with the RATAS building framework model but it is moved further to meet the following requirements:

- To integrate architectural design knowledge,
- To enable semantically meaningful descriptions of buildings,
- To define a frame of representing buildings' information in an appropriate data structure in order to produce coherent buildings (from the architectural point of view).

With respect to the approach of building design and structure, this work focuses to a rather *space-centric* than *construction element-centric* building model. In this way the *DKABM* combines *spaces* that are enclosed by *building elements*. This happens because we consider *form* as the primary element for architectural design, and

secondly because *aesthetic* and *symbolic* intentions are expressed through the morphology of a building.

4.2.2. Framework analysis

The *DKABM* aims to cover all the required information to describe consistently and completely any building seen from the architectural point of view. A building may be described as follows: It is formed by *spaces* organized in *storeys*. A hierarchy of spaces is defined by *subspaces*. Spaces could belong to storeys or can be included in one-to-many storeys. Spaces are formed and bounded by walls, which introduce a boundary to the exterior space and /or environment. The interior space of a building is not merely a single space. It is rather a set of different kinds of spaces. Every space, or group of spaces, satisfies some function(s). These spaces are linked to and are interpenetrated with each other. Adopting the above approach, the description of a building is organised in categories of objects that share same properties and relations.

- The information is organised hierarchically forming a knowledge decomposition tree influenced from *Declarative Modelling* by *Hierarchical Decomposition*. The particularity of our organisation is focused on the fact that we impose a specialised decomposition hierarchy following architectural knowledge. All tree nodes are entities, considered as categories of objects, having properties, inner and inter-relations. The main node of the tree is *Building* containing an abstract description of the building. General information is attached to the building's description, depicting details about the site and the building project. The root node is *Site* that is meant to deal with building code legislation, natural orientation, or other similar functional properties. The *Building* node also contains overall design data such as client and designer information, feasibility studies, versioning details, et cetera. A *Building* can be decomposed to one or many *Building Storeys* and to one or many *Roof systems*. A *Building Storey* gathers information about physical *Elements* and *Spaces*. Elements can be either *Buildings Elements* or *Openings* located on some of them (e.g. a *Door* is located on a *Wall*). At this level, a differentiation is made for (i) the *Structural* and *Non-Structural* building elements and (ii) the *Interior* and *Exterior Openings*. In a *building storey*, *Space* units are enclosed by *Building Elements*. Tzonos [Tzonos 83] proposes an interesting categorization of spaces according to their function. Hence, *Space* comprises the *schemata*, according to the functions of spaces, namely *Public*, *Private*, *Technical* and *Circulation* subspace. In general, a building space is

determined as an assembly of all the above categories of space; this is why the relation among them and *Space* entity, is an aggregation. In Figure 4-2, we also show, as generalization relations, some examples of instances applied to building elements and spaces. Instances are projections of these entities in the domain of Architectural knowledge. We define the entities that participate in the framework and the decomposition and instantiation relations between them.

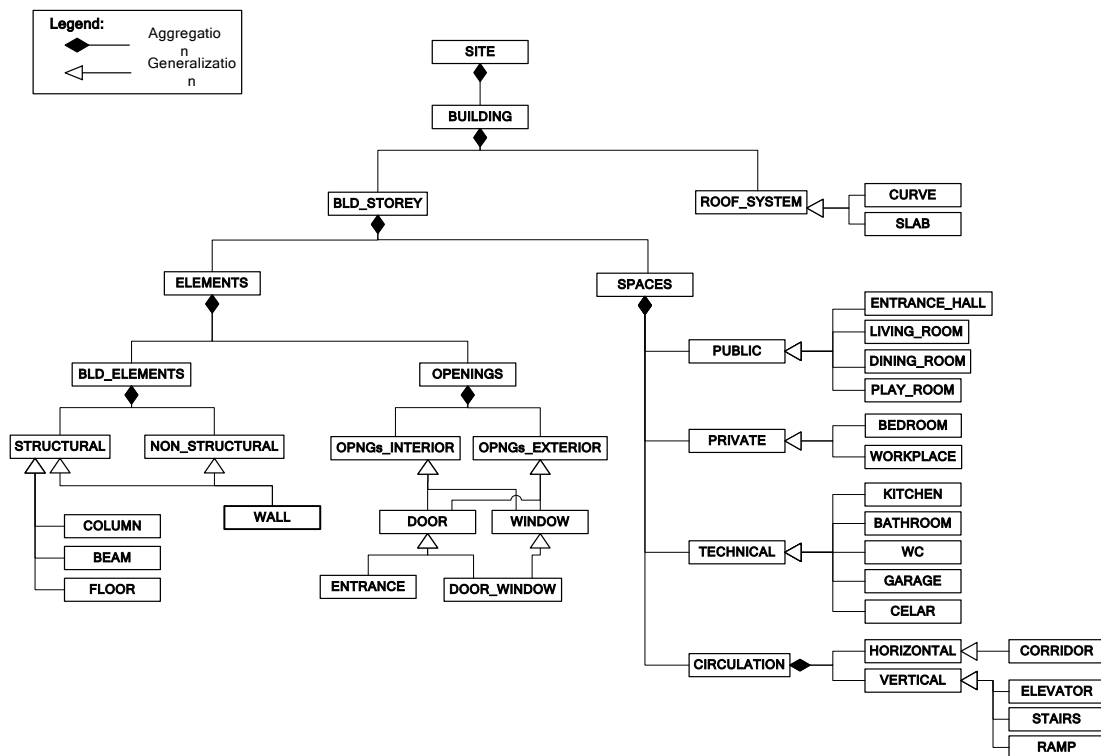


Figure 4-2 DKABM: Entries with aggregation-generalisation relations

Moreover the *DKABM* contains further types of relations and properties in order to describe buildings. Totally, there exist:

- Descriptive and Structural properties, such as the entity's name, its form shape, its dimensions, et cetera.
- Information that describes the function of the entity e.g., a *Kitchen* is used for preparing and eating food. This information is especially useful in further extensions of the *DKABM*. For example, a system can be developed in order to criticise and classify buildings depending on the functions they satisfy or not.

- Declarative properties that are lexical units associated with attribute values: the attributes are evaluated in a specific interval of values, e.g. *Living room Is Large* means that the attribute *Area* varies between 25 m² to 50 m².

Inner-relations, that are logical propositions referred to declarative properties. For example, Living Room is *Longer Than Wide*.

- *Generalisation* relations, who define the instances of entities, e.g. *Living room Is_a Public room*. Furthermore, the structure of the entity is inherited to the instance. For example, as the *Living room Is_a Public room*, it is constrained to have *Minimum_Area* of 10m² (inherited attribute).
- *Aggregation* relations, between an entity and its composite entities e.g., *Living room is part_of Building's space*.
- *Associations* (spatial, topological, et cetera.) between the entities, e.g. *Living Room is Covered_By Roof*. From such associations it is possible to obtain derivative inter-relations for the same entities. For example, *Roof is placed above Living Room*. Extra constraints will be added to the description of both entities in order to enhance the logical integrity of the framework, which is e.g. *Living Room_Area .Roof_Area*. Table 4-1 summarises all the above types of properties and relations combined for the description of an entity in the proposed *DKABM*.

| Properties | | Inner-relations | Inter-relations | |
|------------------------|--|--|-----------------|--|
| Descriptive properties | Object_id, Name | Logical propositions defining Numerical Or Declarative comparisons between the properties of an entity: <i>As Tall As Wide</i> <i>Twice Large</i> <i>Longer Than Wide</i> | Generalisation | <i>Is_a</i> <i>Part_of</i> <i>Adjacent To</i> <i>Axially Symmetric</i> <i>Belongs To</i> <i>Bigger Than</i> <i>Communicates With</i> <i>Covered By</i> <i>Is Enclosed By</i> <i>Larger Than</i> <i>Near To(Entity)</i> <i>In Front Of</i> |
| Function | Utility/Function | | Aggregation | |
| Structural properties | <i>Form,</i> <i>Shape</i> <i>Height,</i> <i>Length,</i> <i>Width</i> | | Association | |
| Declarative properties | <i>Start_Points,</i> <i>End_Points</i> | | | |
| Attributes inherited | <i>Is Tall,</i> <i>Is Large</i> | | | |
| Derived attributes | <i>Minimum_Area</i> | | | |
| General input data | <i>Area, Volume</i> | | | |
| | Remarks | | | |

Table 4-1 Types of Properties and Relations in DKABM

4.2.3. Architectural Constraints

In the *DKABM* framework several mechanisms exist that provide the coherence and the semantic analysis of the proposed description of a building. In order to describe a building in a coherent way, several architectural constraints have to be respected. The constraints are interpreted to restrictions on the existence or not of specified objects, properties and relations in a building description. If these constraints are not satisfied, the mechanisms correct the building description by generating absent (or re-arranging existent) entities, properties or relations, respectively. In this way, two categories of constraints are determined:

- *Obligatory constraints*. A set of constraints obliged to appear in the user's description of the building and if it is not the case, the system generates them.
- *Permissive constraints*. A set of constraints allowed to be applied to each object in case that the user utilises it in the description of the building.

Any other kind of constraints is characterised as *non-permissive*.

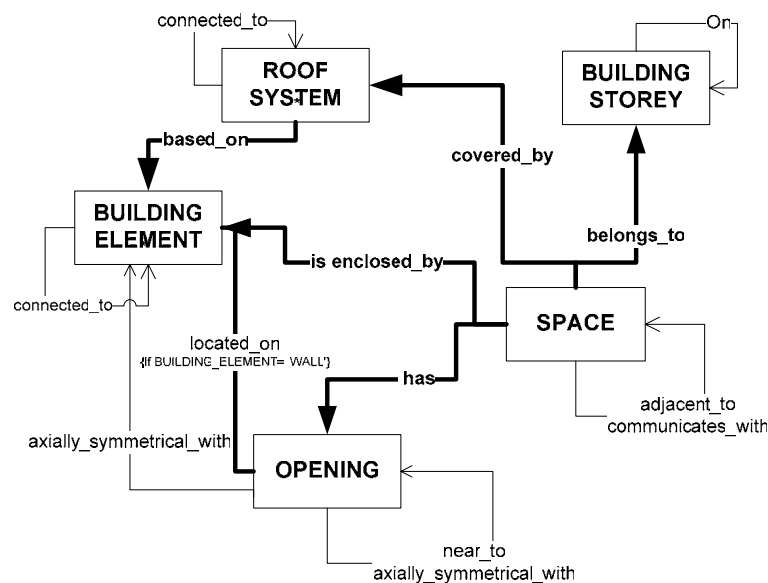


Figure 4-3 Obligatory (highlighted)-Permissive relations (simple arrows) between entities in DKABM

However, different kind of constraints could emerge as soon as they have been introduced as knowledge in the system (either as *obligatory* or *permissive*). Aggregation and generalization relations between entities are characterized as

obligatory, with the purpose of maintaining the decomposition hierarchy. Furthermore, the following nodes are characterized as obligatory because they intercalate for the consistency of the framework:

- The tree nodes Building, Building Storey and Roof system and
- Tree nodes that consist the father nodes of the decomposed objects of the building description upwards until Building node.

For example, from the *Kitchen* node, *Technical_Space*, *Space* and *Building_Storey* nodes have to be generated (Figure 4-2, Figure 4-3) if not yet described. Figure 4-3 illustrates the obligatory and permissive spatial and topological relations between entities. ‘*Space*’ belongs to exactly one *Building storey*. It is enclosed by building elements (such as *Wall*, *Column*) and can be described as having openings (for example, ‘*Kitchen* has an *Entrance* door’). Furthermore, *Space* can communicate to other spaces, directly (defining adjacency) or not. *Roof system* is based on *Building* elements (*Columns* and *Beams*) and covers one or many *Spaces*. Openings are located on *Building elements* (*Wall*) and placed according to specific constraints, such as symmetry (e.g. *the windows are placed symmetrically left and right of the door*), proximity (‘*the window is near the door*’) et cetera.

| LIVING ROOM | |
|-------------|---|
| PROPERTIES | Object_id, Name Utility Form_Shape, Height, Length, Width Start_Points, End_Points <i>Is_Large</i> <i>Minimum_Area</i> Area, Volume <i>Remarks</i> |
| RELATIONS | Is_A(Public Space) Part_of(Space) Covered_By(Roof1), Area=<Roof.Area <i>Longer_Than_Wide</i> <i>Communicates_With(Kitchen)</i> <i>Bigger_Than(Bedroom,2)</i> <i>Communicates_with(Bedroom)</i> <i>Adjacent_with(Bedroom)</i> |

Obligatory properties and relations are written in bold while Permissive ones are in italics

Table 4-2 Definition of Living Room described in ‘*MY_House*’ building

Furthermore, all the above architectural objects can be related with constraints such as comparison ones (*Living Room is Larger than Kitchen*), formative (*Doors has same*

primitive form as Windows), et cetera. According to the above, Table 4-2 presents the definition of Living Room as it is described in the example of ‘*MY_House*’ building. We distinguish obligatory from permissive properties and relations.

4.2.4. Knowledge representation in MultiCAD

As presented earlier, the *Knowledge Base* of MultiCAD provides a precise structure for storing and manipulating the type of elements of the Scene meta-model for different knowledge domains. It is capable of expressing two kinds of definitions:

- The verbal definition of the decomposing objects of a scene (e.g. *Room1 Is_A Living Room*) and
- The formal definition of any constraint (property or relation) applied to them (e.g. *Room1 is Large*. *Large* is a Formative Property, *Room1 is Next_To Room2*. *Next_To* is a Topological Relation).

In the MultiCAD database module, the above definitions are interpreted as an instantiation between the records of the Scene Base and the respective records of the *Knowledge Base*.

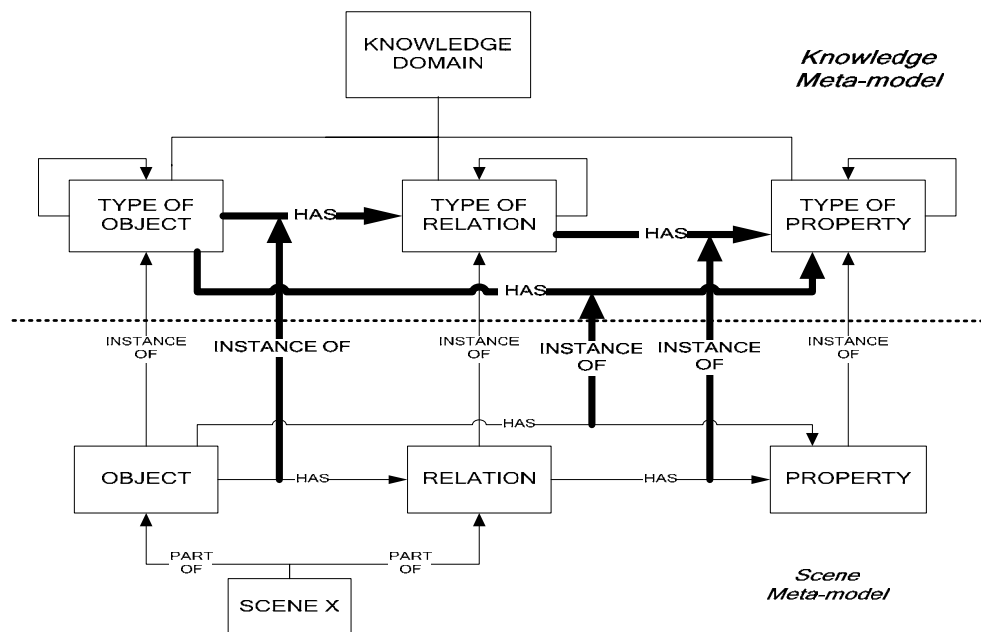


Figure 4-4 Scene and Knowledge *Meta-model* in MultiCAD

(highlighted lines and arrows indicate modifications in relation with diagram in Figure 4-1)

Therefore, MultiCAD architecture deals with the concepts of *Knowledge Domain* (types of objects, relations, properties) without referring to the rules of a domain-specific knowledge. By incorporating the proposed *DKABM* within MultiCAD architecture, we aim towards a domain-specific knowledge system. In order to facilitate the expression of rules between the knowledge domain concepts, we define the *Knowledge Meta-model* updating the *Knowledge Base* of MultiCAD (Figure 4-4) with the architectural constraints of *DKABM*. They are expressed as a relation *HAS* between the tables *TYPE of OBJECT*, *TYPE of RELATION*, and *TYPE of PROPERTY*. This means that the obligatory or permissive constraints are stored in the above tables according to the entities they are applied on. Comparing to Figure 4-1, we note that in this Knowledge meta-model diagram Figure 4-4, the added relations *HAS* are instantiations of the relations between the *Objects*, *Properties* and *Relations* of the *Scene* meta-model [Makris 03].

4.2.5. Normalised Declarative Building Model

Architectural knowledge is introduced within MultiCAD, using the above *Knowledge Meta-model*. In particular, we introduce in the Knowledge Base the structure of *DKABM* concepts along with their properties and relations, defining the knowledge domain of architecture in MultiCAD. This implies that *DKABM* entity information is interpreted to records in the MultiCAD tables of the *Scene Base* and the *Knowledge Base* and extra tables are added to support the domain of values. Any scene of Building described in MultiCAD must follow this structure of architectural knowledge. In case that a scene has an incomplete description or a invalid one, a mechanism is provided to correct this description either by re-arranging the scene decomposition tree or by generating instances of objects according to the missing concepts. Hence, the *Normalized Declarative Building Model (NDBM)* is the scene representation of MultiCAD, enhanced with architectural knowledge through the *DKABM*. The description of the ‘*MY_House*’ building is re-applied on MultiCAD and enhanced via *DKABM*. The steps followed are:

- The designer describes a scene of a building defining, also, the type of the decomposing objects and the type of their relations and properties. The designer requests this building to conform to the description of a typical dwelling. Figure 4-5 shows a representation of ‘*MY_House*’ based on the *EE-R* model.

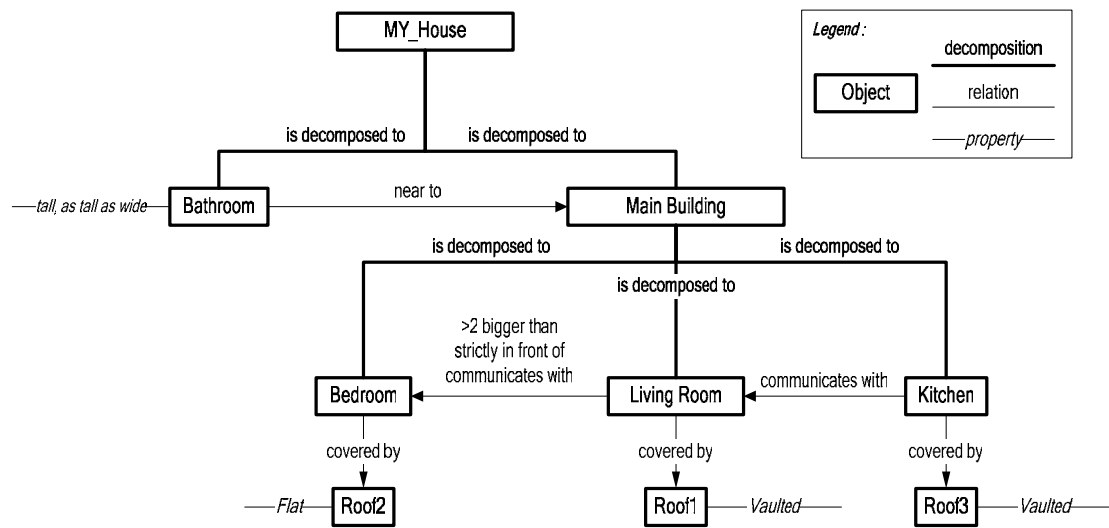


Figure 4-5 Representation of 'MY_House'

- The system applies the mechanisms of coherency, correcting the input description to match up with the decomposition hierarchy of the *DKABM* for the typical dwelling. As a result, the *Normalized Declarative Building Model* of the scene is produced.
- The normalized description of the building is stored in the *Scene* and the *Knowledge Base* of MultiCAD.
- The *Internal Declarative Representation (IDR)* of the scene is updated according to the produced *NDBM*.

It is obvious that missing obligatory entities have enforced the generation of respective objects, relations and properties.

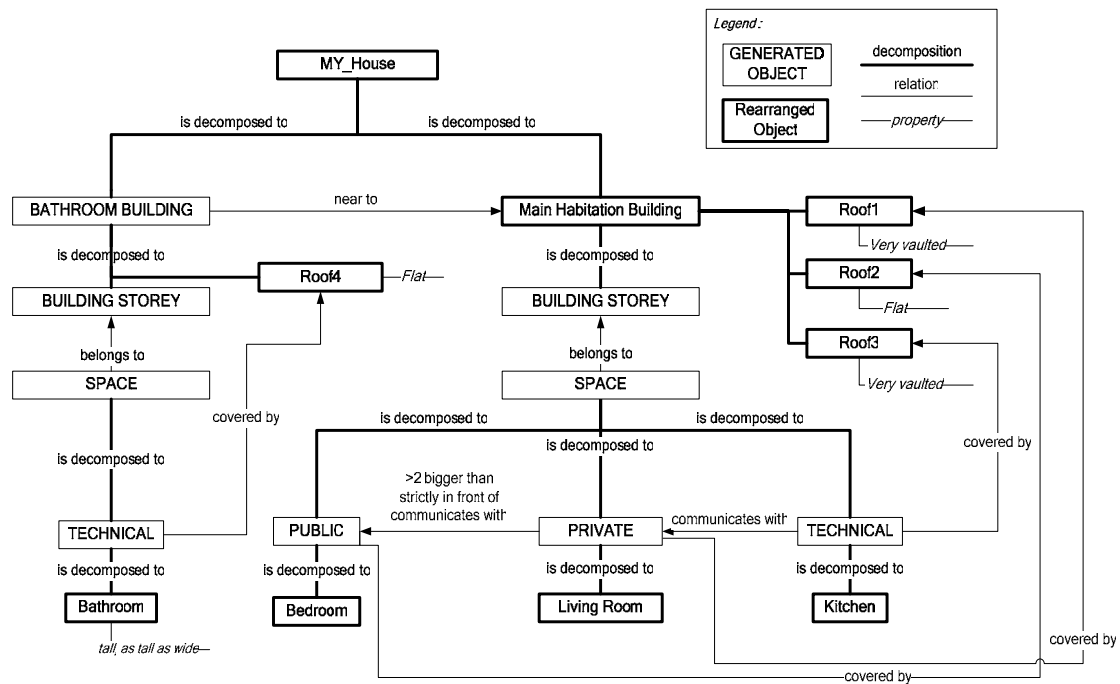


Figure 4-6 Normalized Declarative Building Model of 'MY_House' habitation

As it is shown in Figure 4-6 the existing decomposing objects of Figure 4-5 have been re-arranged according to the generated ones. We note also, that a *Roof* object has been generated for the *Bathroom* object in order to keep the consistency of the *DKABM*.

4.3. Architectural style in MultiCAD

In this section the main focus of the research is to develop a model that is rich enough to capture and express qualities of building representations, (relationships - properties), that are important in reasoning with stylistic design principles. In this respect, within a declarative design system, we introduce the concept of 'style' as it is defined in the Chapter 3. Our task is directed towards the introduction of a framework for the description of architectural style. In this way the designer will have the ability to decide which features-properties of style to be introduced in a declarative design environment for the adaptation to a particular style, possible evolution of a style and the emergence of new styles.

4.3.1. Modelling scheme

In order to develop improved *Computer Aided Architectural Design (CAAD)* environments it is of great importance the modelling of architectural design knowledge, and, in particular, that part of knowledge concerning architectural style(s). Geometric representation alone cannot provide support at the conceptual stage design tasks. During conceptual design, it is not possible to conclude towards geometrical solutions, as this provokes an explosion of solutions. In addition, they are not too precise at this design stage. Conceptual designs are more judicious in a first stage, they can be compared to architects' sketches and working models in this primary research of placement principles. We consider a particularly challenging aspect for the integration of architectural design knowledge: how to develop a general structure for representing an architectural style that responds to two issues:

- Stylistic information must be represented independently of any style category. Style representation must follow a frame, which can be easily adapted among the different architectural styles, and incorporate all different knowledge of styles.
- The representational frame must ensure a full representation of the semantics of style.

Our main intention is the development of an informationally complete and semantically fertile building's style representation. Such a representation is competent in supporting both the interpretation and the evolution of architectural styles within a declarative design system. This representation is based on the following two principles:

- We address only one phase in the overall design process. In specific, the focus is on conceptual design phase of architectural design.
- We make a conscious effort to integrate the representation of spatial, geometric and structural characteristics in one unified representational frame.

4.3.2. Style Knowledge in MultiCAD software architecture

We describe the formation of a *Style Library*, which provides mechanisms for storage and update of the knowledge of specific architectural styles. Apparently, such a library should be contained and fully take advantage of the mechanisms within the knowledge base of MultiCAD. Furthermore, it provides the appropriate procedures in

order to ‘*translate*’ any possible stylistic constraint in the form of an objective function.

4.3.2.1. Style framework

A new important element of a *Style Library* is the need for a set of principles that formally define each architectural style. The spatial composition of specific type of objects is crucial for the expression of architectural style principles (Chapter 3). In detail, firstly, certain types of properties are important for the expression of style. Secondly, another fact for the accurate expression of a style is the topological relationships between objects. In order to represent stylistic principles within the declarative modelling cycle of MultiCAD, we take advantage from the introduction of the *DKABM* framework in MultiCAD. We employ the distinction in the appearance of constraints in the declarative description of a scene (building). So far such a distinction has not appeared in the applications of dedicated declarative modellers. Therefore we consider on the one hand any constraint of the initial scene description as hard constraint that should not be violated during the generation of alternative solutions. On the other hand, the architectural style that the user prefers for his/her description to adapt as *soft constraints*. We will explain how this distinction will be introduced in the description and in the generation phase of a declarative modeller such as MultiCAD.

4.3.2.2. Style representational scheme

We introduce an enriched modelling framework for architectural style that will be based on the architectural design knowledge and the *DKABM* framework within MultiCAD system architecture. Secondly, we provide updated and design oriented geometric entities capable to deal with the demanding representation of architectural designs. There exist many cases in the literature, which stated that it is difficult the existence of a geometric configuration for design methodology which can fulfil the following requirements all at once: modelling with very simple modelling entities. Entities suitable to the vague geometric information that exist in the conceptual design stage, Entities related together by constraints and parameters. The intention of this section is to describe style layout modelling methodology, based on the mentioned requirements suited to the needs of performing compositional synthesis between

abstract geometrical elements. The layout of an architectural style can be interpreted according the following definition:

The layout of a style is a set of its compositional components, represented by abstract geometric elements embodied by the overall shapes and dimensions. The elements of the layout are composed together by means of stylistic principles (in the form of objects and constraints), which determine the architectural character of the layout. The fulfilment of this approach necessitates the introduction of the subsequent description concepts. The abstract elements used for representing spatial and building elements are called *geometric entities*. The dominant feature of a representational scheme for architectural style, is a basic geometric entity around which all the above three categories are build. There is an obvious need for a representation as a basis that will define comprehensively:

- What constitutes the topological relationships,
- What constitutes the formative conditions,
- What constitutes the building typology.

In that scheme it is of equal importance how the interrelations among the three categories are defined. In the subsequent parts of this section, first we explain the data scheme of the various elements that constitute the above representational scheme, with reference to a simple geometric entity. Second we apply that scheme to the proposed two example architectural styles, that of Santorini and Metsovo habitation.

4.3.2.3. Defining semantics for architectural style

We present a framework within which semantics is effectively incorporated in the definition of architectural style(s).

- A variety of constraint types is defined, each of which can capture specific aspects-characteristics of stylistic intent (stylistic principles).
- A class of features is defined, integrated with the constraints.
- A formal geometric definition for a semantic stylistic model is presented, providing the basis for a comprehensive definition of different architectural styles.

Constraints offer a convenient means to specify properties and characteristics, which the designer requires in a style model.

4.3.2.4. Stylistic constraints - properties

Three types of constraints are used to specify stylistic characteristics in and among elements of a style: topologic, geometric and algebraic constraints.

- *Geometric* constraints specify geometric relations (e.g. parallelism, perpendicularity, et cetera.) between style elements. Geometric constraints can be used to specify geometric properties-characteristics either within an object instance, or between sets of different object instances. An example of the former is the characteristic '*the side faces of living room and kitchen should be parallel*', and an example of the latter is '*the axes of two holes should be at a specified distance*'. An essential property of geometric constraints is that they operate on elements of architectural styles, their faces, starting points.

- *Algebraic* constraints: specify expressions like equalities or inequalities, among elements parameters. Algebraic constraints can be used to specify parameter relations within an element, e.g. equality between the width and the length of a space, or among parameters of different elements, e.g. '*the length of a living-room should be twice the length of a bedroom*'. Additionally algebraic constraints are used for the specification of the allowable range of values for an element parameter, by means of inequality relations.

- *Topologic* constraints: Such constraints specify relations concerning the relative position of stylistic objects (spatial and/ or structural). Topologic constraints are specified on elements of an architectural style by using the topologic relations. Such topologic relations are the following: *Near_to*, *Adjacent_of*, *Above_of*.

4.3.2.5. Definition of objects

Stylistic objects can be defined as representations of form aspects of different parts of a building, and are significant for the definition of an architectural style. Each element has a well-defined meaning, expressed through its topologic, formative and building properties. Form and parameters of elements are specified using stylistic constraints. Each stylistic element could be represented as parameterised object. A basic object encapsulates a set of geometric constraints that relates its parameters to the corresponding object. The geometry of an element accounts for the bounded region of space comprised by its volumetric form. Moreover, its boundary is decomposed into functionally meaningful subsets, its faces. Each face is labelled with its own generic

name, to be used in design-compositional operations. For example, a cuboid has a top, bottom, east, west, north and south face.

4.3.3. Framework analysis

The *DKABM* framework as already implied could cover all required information to describe consistently and completely a building seen from architectural point of view. We follow the same hierarchically organisation of information forming a knowledge decomposition tree influenced from *Declarative Modelling by Hierarchical Decomposition*. It is the same specialized decomposition hierarchy following architectural knowledge as adopted in the previous section of this chapter. The main node of the tree is ‘*Site*’ containing an abstract description of one or many buildings. The ‘*Site*’ deals with possible initial design directions (i.e. building code legislation, natural orientation, or other similar functional properties). ‘*Building*’ is decomposed to one or many building spaces and to one or many systems of ‘*Roof(s)*’. The role of constraints in declarative modelling is very significant. A complex scene could be described without difficulty with the use of constraints. Any relations between objects as well as properties of objects could be specified by the declarative semantic of constraints.

In this thesis we will employ an important change on the above approach, because of two reasons. Firstly, during the architectural design process all constraints are not of the same importance. Therefore the achievement of an architectural design oriented declarative design system it is needed the distinction between constraints. Secondly, in order to represent architectural style it is fundamental to use and manipulate constraints. In particular, we utilise constraints not only for scene description, but in advance for the expression-description of architectural style principles. The achievement of such effort makes necessary the distinction of constraints as *hard* and *soft*.

The current application is mainly focused on the distinction between these two types of constraints. As we have already argued in Chapter 2, stylistic constraints belong in *soft* constraints, in general. Therefore in order to employ stylistic principles for the generation of building designs adapted to architectural styles, we provide the distinction of *hard* and *soft* constraints where needed in the *Knowledge Database* of MultiCAD. Such separation is provided for the first time in a declarative modeller. In

particular the three tables of *Type of Objects*, *Type of Properties* and *Type of Relations*, would incorporate specific information as explained in the following.

The table of *Type of Objects* contains a number of specific objects as architectural spaces. For the reason that in the current stage of our research we concentrate on the architectural style of habitations the contained objects are the following: *Kitchen*, *Bedroom*, *Living room*, *Dinning Room*, *Office*, *Guest Room*, *Bathroom*, *WC*, *Storage*, *Garage*, *Corridor*, and *Hall*.

The table of *Type of Properties* contains fundamental information for the description of the scene objects. This table has many properties, for example *Placement* (with a range for minimum-maximum coordinates), *Length*, *Width* and *Height* of objects. Their values could be either numerical and / or declarative.

The table of *Type of Relations* contains relations in the form of the two distinct types of constraints. The relations representing *hard* constraints define the description of a scene (building). Such relations should not be violated during the generation of alternative solutions. The relations representing *soft* constraints play an important role for both the representation of style and during the generation of solutions. In the first case, these relations are fundamental for the expression of the principles of an architectural style. In the second case, these *soft* constraints will play the role of *objective criteria* for the evaluation of the evolved solutions by the genetic algorithm during the generation phase.

4.3.3.1. Spatial planning constraints

4.3.3.1.1. Hard constraints

Given the abovementioned distinction we obtain as *hard* constraints the following binary relations for the arrangement of the spatial objects of a description. Two elements could be adjacent on one of their six faces, (North-South-East-West-Above-Below). In order to enable the circulation between the two spaces we consider a minimum distance of adjacency. There are the following, *Adjacent_North*, *Adjacent_South*, *Adjacent_East*, *Adjacent_West*, *Adjacent_Above*, and *Adjacent_Below*. Two elements could be near each other on one of their six faces, (North-South-East-West-Above-Below). Furthermore, it is considered a minimum distance of nearness. There are the following, *Near_North*, *Near_South*, *Near_East* and *Near_West*. (Details of formulas in Appendix Definition of Constraints)

Two elements could overlap on one of their four faces, (North-South-East-West). A minimum variable distance of overlapping could be considered between the two spaces. There are the following, *Overlap_North*, *Overlap_South*, *Overlap_East*, and *Overlap_West*.

A set of six relations that concern object's placement within the site. These are: *Place_North-East*, *Place_North-West*, *Place_South-East*, *Place_South-West*, and *Place_Centre*. The relation *Inside* forces placement of an object within the boundaries of another object. (Details of formulas in Appendix Definition of Constraints)

4.3.3.1.2. Soft constraints

We obtain as *soft* constraints a number of binary and unary relations. Considering the geometric properties of the spatial objects we have, the following relations are defined:

Binary relations: *Same_Length*, *Same_Width*, *Same_Height*, *Same_Area*, 'Same_Volume'.

Unary relations, *Longer_than_Wide*, *Longer_than_High*, *Wider_than_Long*, *Wider_than_High*, *Higher_than_Long*, *Higher_than_Wide*.

For the relative placement of the objects we have the following binary relations: *Axial-Centre_North-South*, *Axial-Centre_South-North*, *Axial-Centre_West-East*, *Axial-Centre_East-West*.

Axial-Length_East-South, *Axial-Length_East-North*, *Axial-Length_West-North* and *Axial-Length_West-South*.

Axial-Width_North-East, *Axial-Width_North-West*, *Axial-Width_South-East*, and *Axial-Width_South-West*.

The constraint *Align* is defined as adjusting objects position in relation to a certain line. Parameters for the *Align* operator are the alignment line and edges of objects to be aligned. Also, objects locations can be adjusted with regard to any part of these elements, i.e. edges, centres, or axis.

Align-Length_East-South, *Align-Length_East-North*, *Align-Length_West-North* and *Align-Length_West-South*. *Align-Width_North-East*. *Align-Width_North-West*, *Align-Width_South-East* and *Align-Width_South-West*. All these constraints could be applied both on the building as *global* constraints, and /or the spaces as *local* constraints. (Details of formulas in Appendix Definition of Constraints)

We should underline the fact that in many research projects [Bonnetfoi 99], [Fribault 04] they appeared the same type of relations. However, their application in our research is differentiated in the following points:

- Firstly, there exists a separation between *hard* and *soft* constraints, rather than utilised as hard constraints in the beginning of the description.
- Secondly, they are utilised as evaluation criteria. The benefit of such an approach is it is possible with the same *soft* constraints the expression of a variety of styles.

4.3.3.2. Aesthetic evaluation

In order to capture the aesthetic intentions of the designer we also introduce a number of aesthetic criteria in the form of *soft* constraints. Such aesthetic criteria also applied for style evaluation: *Balance*, *Equilibrium*, *Unity*, *Density*, *Regularity*, *Homogeneity*, *Rhythm*, *Symmetry* (vertical-horizontal), and *Proportion* [Ngo 02], [Staudek 02] (Details of formulas in Appendix Definition of Fitness).

- *Balance* can be defined as the distribution of optical weight in a design layout (plan, elevation, and facade). Optical weight refers to the perception that some objects appear heavier than others. Larger objects are heavier, whereas small objects are lighter. Balance in design layout is achieved by providing an equal weight of design objects, left and right, top and bottom. It is computed as the difference between total weighting of objects on each side of the horizontal and vertical axis.
- *Equilibrium* is stabilisation, a midway centre of suspension. It is computed as the difference between the centre of mass of the displayed objects and the physical centre of the bounding box.
- *Unity* is coherence, a totality of objects that is visually closely all in one piece. In planning design it is achieved by using similar dimensions and leaving less space between objects of a design than the space left at the margins and within the objects. The objects are dimension-related, grouped together and surrounded by white space. (If no specific initial space layout is defined, then the layout and bounding box is the same).
- *Density* is the extent to which a specific space layout (or the resulting bounding box of a set of objects), is covered with objects.

- *Regularity* is a uniformity of spatial objects based on some design principle or plan. In architectural design it is achieved by establishing standard and consistently spaced horizontal and vertical alignment points for spatial objects, and minimizing the alignment points between them.
- *Homogeneity*. The relative degree of homogeneity of an architectural composition is determined by how evenly the objects are distributed among the four quadrants of the plan. The degree of evenness is a matter of the quadrants that contain more or less nearly equal numbers of objects. It is by definition, a measure of how evenly the objects are distributed among the quadrants.
- *Rhythm* in design refers to regular patterns of changes in the elements. This order helps to make the appearance exciting. It is accomplished through variation of arrangement, dimension, number and form of the objects. The extent to which rhythm is introduced into a group of elements depends on the complexity (number and dissimilarity of the elements).
- *Symmetry* is axial duplication: an object on one side of the centre line is exactly replicated on the other side. Vertical symmetry refers to the balanced arrangement of equivalent objects about a vertical axis, and horizontal symmetry about a horizontal axis. By definition, it is the extent to which the spatial layout is symmetrical in two directions: vertical.
- *Proportion*. In architectural design, aesthetically pleasing proportions should be considered for major objects of the design, including spaces. '*Proportion*', is the comparative relationship between the dimensions of the design objects and proportional shapes.

4.3.3.3. Roof morphology constraints

4.3.3.3.1. Hard constraints

In the case of the roof morphology we have as *hard* constraints the following binary relation for the definition of roof in particular building composition: The relation *Covers* in order to declare that one roof will cover from one to many spatial objects. With this constraint a roof inherits the dimensions of a space. In the case of more than one spaces the roof inherits the dimensions of the resulting bounding box that these

spaces form. At this stage of our research we do not utilise any other hard constraint for the roof.

4.3.3.3.2. Soft constraints

In order to obtain the characteristic roof forms which belong to particular styles we define a number of new *soft* constraints. All these constraints control the geometric parameters for the formation of a roof. Their application is for a specific roof each time. However, they should be repeated according to the number of the roofs in a building design. They differentiate in two categories. The first category contains these which control the consistency of the roof morphology, *Roof_Homogen*, *South-Side_Homogen*, *North-Side_Homogen*, *East-Side_Homogen*, *West-Side_Homogen*, *Parallel-to_Longer-Dim*, *Parallel-to_Shorter-Dim*, *Rectangular_Base*, *Circular_Base*, *Roof-Center_in_Center*. The second category contains constraints that control the degree of curvature, linearity, cavity of the roof form. In general, they define three control points of the eight curves that constitute the roof form. They are the following: *Minimize_Triangle-Area*, *CoLinear*, *W1W2-Line_Parallel-to_horizontal*, *W2W3-Line_Parallel-to_Vertical*, *Maximize_W1W3-Length*, *W1W2_EqSqr*, *Minimize_W1W2-Length* and *Minimize_W3W2-Length*. Details of formulas appear in Appendix Definition of Constraints.

4.3.3.4. Measure of Style

A style could be expressed as a set of objective criteria. We express the measure of style (degree of adaptation to a style) as an aggregate of these objective criteria for a scene spatial planning composition. A stylistic scale is created, with total adaptation (i.e. a scene is 100% adapted to a particular style) at one end, and lack of adaptation at the other (i.e. a scene is 0% adapted to a particular style). The general formal expression of the evaluation is given by:

$$SE = g\{f_i(E_i)\} \in [0,100] \quad (1)$$

The (E_i) (with $i= 1, 2, \dots, n$) is the set of n evaluation criteria that express a specific style. The $f_i ()$ is a function of E_i and is functionally related to the evaluation criteria that characterise $g \{ \}$. In particular, the function $f_i ()$ expresses the local metrics of each participated evaluation criterion. With the utilisation of f_i we illustrate how each

objective criterion in the proposed evaluations contributes to the overall evaluation of a scene for a style. The function $g\{ \}$ is an intergrading function in an n -dimensional space, where the n depends upon the number of soft constraints that participate in the expression of a style. In order to express the style of a building spatial composition we imply the calculation of the sum of the number of the stylistic objective criteria. Therefore equation (1) will obtain the form of a linear summation of the weighted evaluations.

$$g\{ \} = \sum_i^q a_i f_i(E_i), \quad 0 \leq a_i \leq 100 \quad (2)$$

For each objective criterion (stylistic *soft* constraint) a constant weighting factor a_i is provided. In the beginning all objective criteria have a weighting component set to 1, and they have equal importance. Later, during the formation of a combination of objectives for a style, their weight factors are changed in accordance with the demands of the expressed style. The process for the determination of the weight factors we will further explained in details in the following parts of the current chapter.

4.3.4. Style in Description phase

Given the earlier description the introduction of style within the description phase is applied as follows. The designer could provide a scene description; the applied architectural design knowledge in the form of *DKABM* provides specific type of objects, type of properties and type of relations for the description of a potential scene by the user. The *DKABM* mechanism guaranties the validity of the scene description; otherwise it could rearrange the structure of the description, and while at the same time corrects its validity according to the embedded architectural knowledge. So far the description contains only the preferred objects, their properties and possible relations in the form of hard constraints. The following step presupposes the application of the genetic algorithm as the generation engine of solutions. Given this fact, then the designer could select a particular style. A style is defined as a combination of soft constraints applied over the spatial objects and/or the whole building. By the completion of that step the user could select to move to the next step of the declarative cycle, the generation of scenes.

4.3.5. Style in Generation phase

During the generation phase the introduction of style does not affect the initial scene description. A specific style module provides combination of soft constraints. These constraints will be translated in the form of objective fitness function for the evaluation of the generated scenes by Genetic Algorithm generative engine.

4.4. Geometric level of representation

We provide updated and design-oriented geometric entities capable to deal with the demanding representation of architectural designs. We make use of a three-dimensional design environment, therefore spatial and building elements will be expressed as three-dimensional objects expressed through polyface-mesh geometry. For the needs of style representation we utilise two types of geometric entities. One for the *Spatial* object and one for the *Roof* objects. The later are more complex objects with special properties and would be capable to obtain many alternative forms.

4.4.1. Spatial object geometry

For simplicity, this representational scheme assumes that cuboids or combinations of quadrilaterals can represent all spaces. This simple representation can model a large array of spatial elements. For the spatial objects we make use of an isothetic box. Such entity has an insertion point on the lower left vertex, (Figure 4-7).

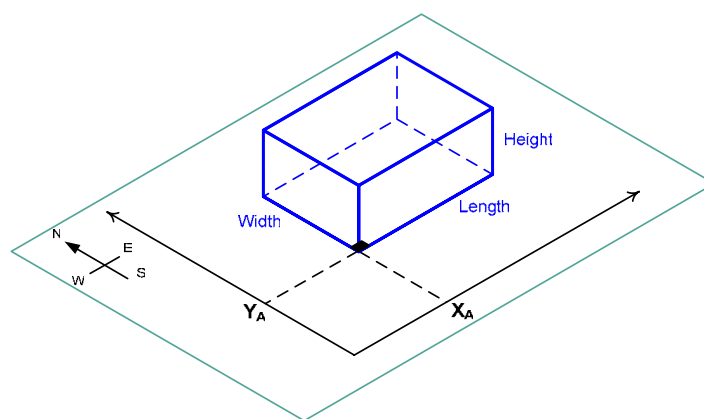


Figure 4-7 Isothetic box

Such a box is defined using six numeric variables: the three coordinates of its placement point (x , y , and z) and by a displacement vector defining the dimensions of

the associated bounding box (length, width and height). The main characteristic is that its faces are parallel with the planes of the absolute coordinate system.

4.4.2. Roof object geometry

For the roof objects we introduce a different geometric entity. We define a representation that enables both linear and curvilinear roof objects in order to make possible the appearance of alternative forms for the roof object. While the geometric representation is a little bit complex, however the specific proposal allows the designer to have a total control over its deformation procedure(s). Moreover this representation does not increase the problem dimensionality, while it offers flexibility in the formation of alternative complex roof forms. We consider a three dimensional parallelepiped which, except for its base face, the rest of its faces are defined by eight *splines*. Each *spline* is defined within An orthogonal frame formed in the *XZ* plane of the coordinate system (Frames: *A, B, C, D, E, F, G, H*) (Figure 4-8).

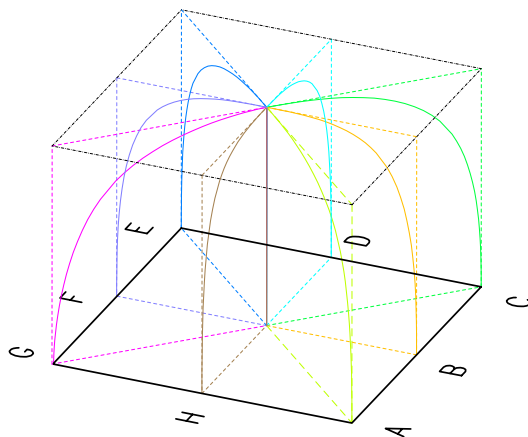


Figure 4-8 Frames A-B-C-D-E-F-G-H

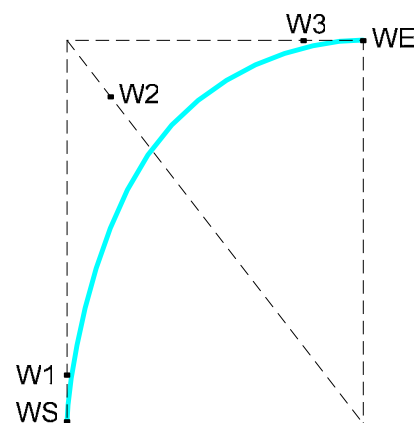


Figure 4-9 Fitting points in a frame

Each of the *splines* has five *fitting points*. The *start point* (*WS*) and the *end point* (*WE*) are fixed at the lower and the upper end of the frame respectively (Figure 4-9). The three *fitting points* (*W1-W2-W3*) move within the limits of the frame. In this way they could provide variable degrees of *curvature*, *cavity* and *linearity* for the shape of the roof. In particular, *W1* and *W3* move only on the vertical (up and down) and horizontal (left and right) side of the parallelepiped frame.

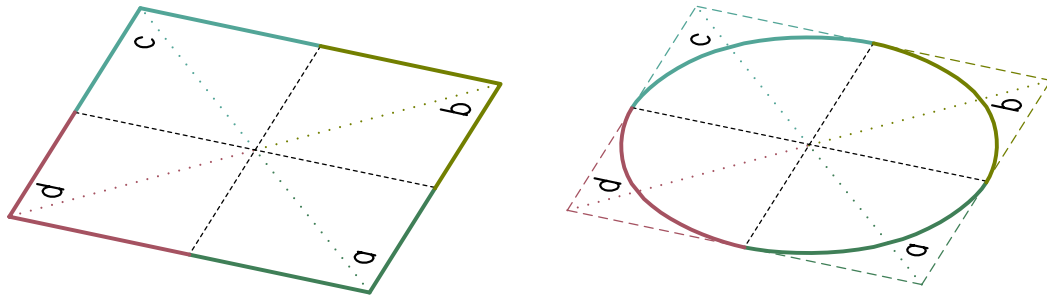


Figure 4-10 Frames a-b-c-d

The base face of that entity is defined by four *splines* (*a-b-c-d*) which could provide two shapes for the base of the entity either circular or parallelogram, (Figure 4-10). With this two alternative base shapes we could obtain almost any possible roof shape among the total amount of architectural styles.

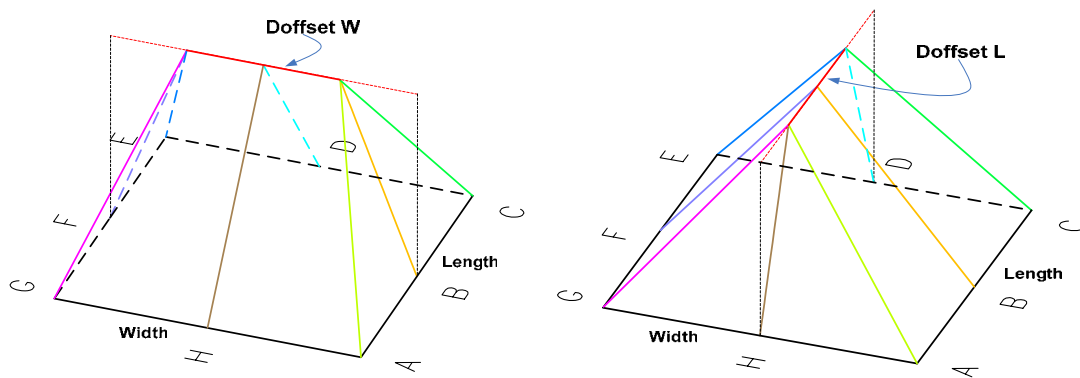


Figure 4-11 $D_{OFFSET} Length - D_{OFFSET} Width$

In order to have accurate control over the formation of a roof we define four more variables for the geometric entity. First, D_{OFFSET_Length} , D_{OFFSET_Width} , control the magnitude of the ridge of the roof, (Figure 4-11).

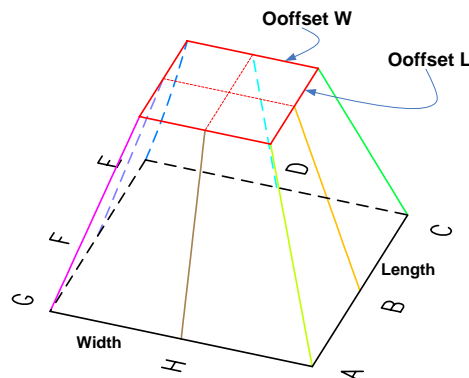


Figure 4-12 $O_{OFFSET} Length - O_{OFFSET} Width$

Second, O_{OFFSET_Length} , and O_{OFFSET_Width} control the magnitude of the flat top or the ridgeline of a roof (Figure 4-12). These variables control basic aspects of the roof shape. They follow some examples of possible roof forms as provided by the geometric entity (Figure 4-13).

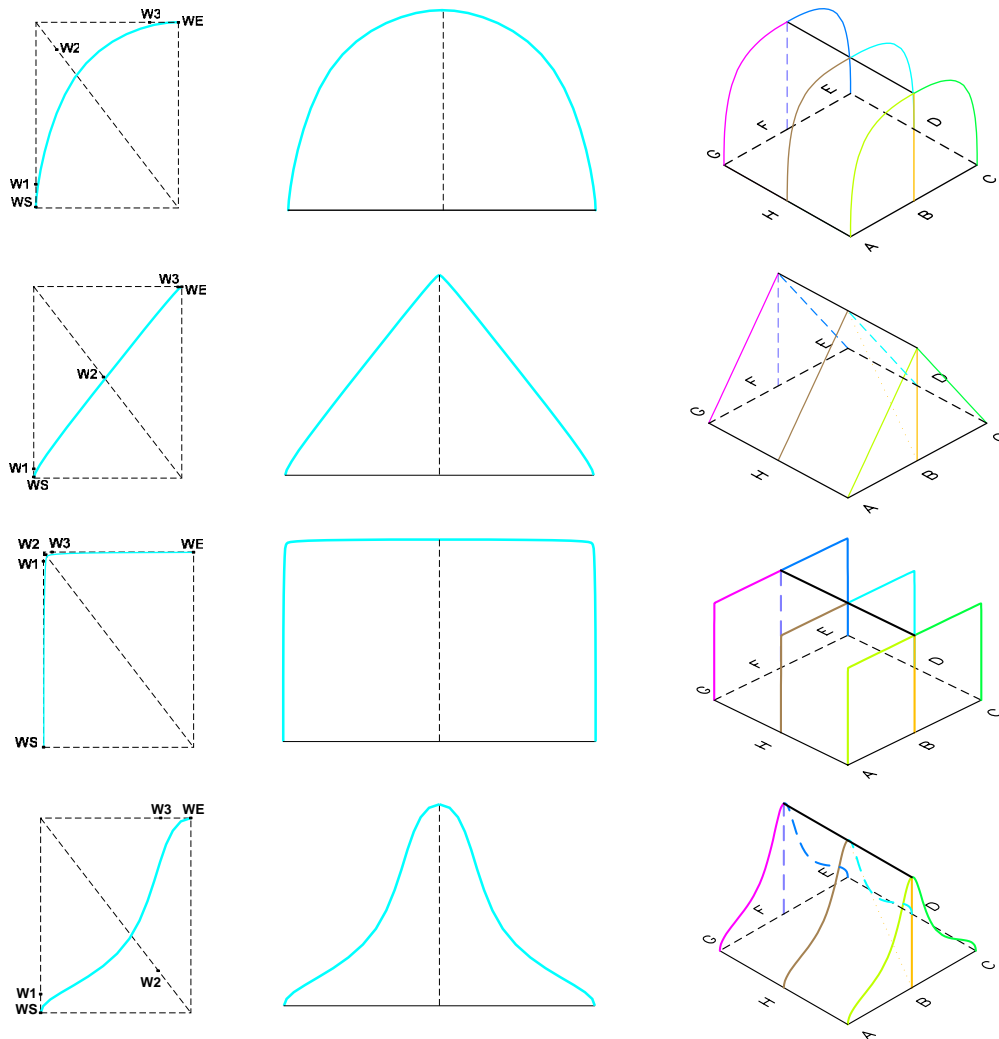


Figure 4-13 Examples of roof forms

4.5. Simple Genetic algorithm

The first approach for a genetic algorithm-based declarative system is described. This system constitutes a first attempt to build an evolutionary declarative design system. In the beginning we argue for the choice of GA, as a fruitful generation engine. Specifically, we present, how the terminal subscene bounding boxes of the internal

tree model are encoded into chromosomes, which genetic operators are used and in what way and, finally, how new generations of solutions are reproduced and evaluated by the user. A simple genetic algorithm (*SGA*) is used as a feasible generation engine mechanism within MultiCAD system. The *SGA* is the basis for the proposed declarative design generative process, as shown in. The successful formation of a *GA* program needs the mutual use of appropriate data structures that correspond to the chromosome representation, and appropriate algorithms that correspond to ‘*genetic*’ operators that transform one or more individual chromosomes. We make use of a genetic algorithm based on the *SGA* [Goldberg 89]. The genetic operators are based on the consideration of needs of architectural conceptual design. The modification concerns the initialisation of the system for the generation of first generation. The application of a *SGA* to a domain-specific problem considers the development of four fundamental elements. First, the phenotype must be indicated. The phenotype would form specification and enumeration of the search space, which will define the permissible solutions to the problem. Second, definition of genotype as a form of encoding the solutions. Third, *GA* must be determined given the demands and the characteristics of the problem. Fourth, the formation of fitness function(s) will enable the evaluations of prospective solutions of the problem for the *GA*.

4.5.1. Phenotype

The definition of scenes-designs as phenotypes is based on a simple geometry representation, which is formed with the use of surface modelling geometry. A surface object is made by wire-frame geometry and algebraic equations to define an area between the edges of the object thus producing its surfaces.

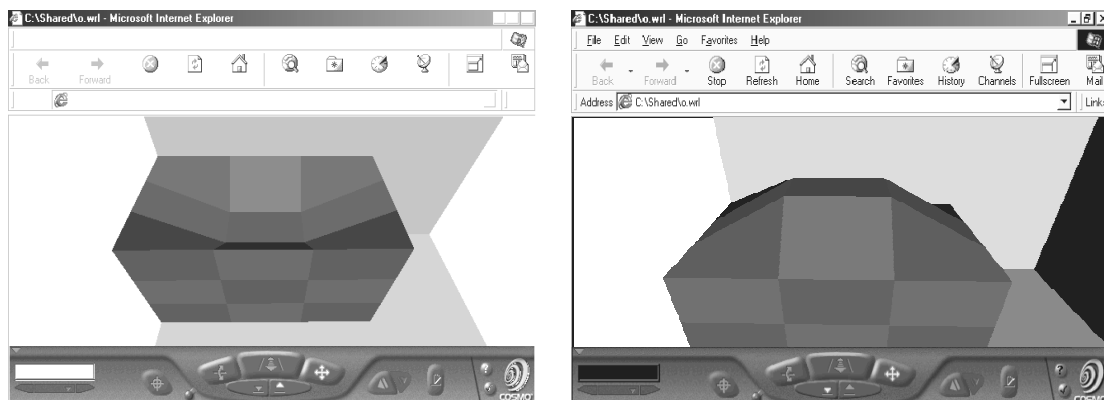


Figure 4-14 Examples of basic volumes

In particular, we use a general polygonal mesh, which it is capable to approximate curved surfaces (Figure 4-14). The generic geometric object is a quadrilateral made by six surfaces. Each one of the surfaces is composed by a polyface mesh of 3 by 3 faces, (Figure 4-15).

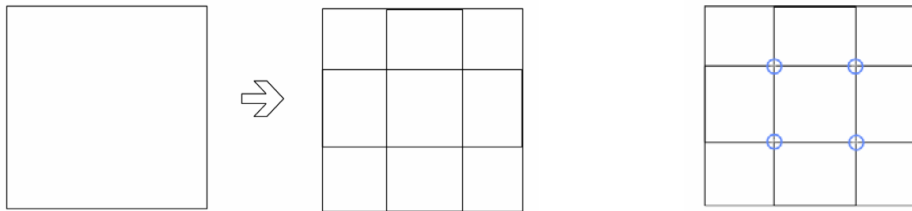


Figure 4-15 Polyface mesh

This geometric object requires six parameters for its geometric definition. In turn, building designs are defined by a number of geometric objects, for both spatial and structural elements. A phenotype of a design is a flat list of ordered parameters that define the geometry of every basic object. Figure 4-16 shows a list of parameters of a building evolved phenotype.

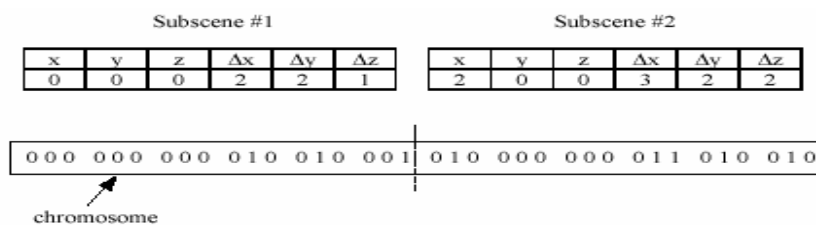


Figure 4-16 Chromosome representation of a scene with two subscenes

4.5.2. Genotype Chromosome Encoding

The genotype consists of a linear chromosome. Multiple blocks of six genes, and each gene being defined by 18 alleles compose the chromosome. Figure 4-16 presents an example of a chromosome. Each of the 3D scenes recursively consists of 3D subscenes. As explained above, the scene nodes of the global scene description tree can be considered to be contained within bounding boxes that satisfy a set of constraints imposed on their positions and relative dimensions. The bounding box can be described by the bottom-front-left corner (x, y, z) relative to the global (X, Y, Z)

coordinate system and by the three dimensions Δx , Δy and Δz for its width, depth and height respectively. The origin of the global coordinate system is assumed to coincide with the bottom-front-left corner of the bounding box that corresponds to the root node of the scene description tree (the global bounding box). A representation of the integer quantities is as binary string encoding. The length of these binary strings will depend on the required accuracy as well as on practical issues such as, the induced time complexity on the solution generation engine. A chromosome representation of a scene must contain all spatial information regarding the terminal nodes (subscenes) of the scene description tree. Assuming that a scene contains n such nodes, a chromosome will be represented using nL bits. Figure 4-16 shows the encoding for a scene of two subscenes using $L = 3$. The dotted line signifies the end of the first subscene representation and the beginning of the other.

4.5.3. Evaluation functions

During this stage the SGA does not have fitness functions to obtain information from the evaluation of the degree of performance for the individuals of every generation. In particular, only the designer selects on every generation the individuals he/she considers the best.

4.5.4. Selection mechanism

For selection we use proportional or biased roulette wheel selection was used.

4.5.5. Genetic operators

The genetic operators used for reproduction of the current generation of solutions are: cloning (elitism), crossover and mutation. The three operators are applied following a user evaluation of all chromosomes. The Crossover operator requires two parents. Each chromosome is given a probability to be selected equal to the ratio of its fitness score to the sum of the fitness scores of all chromosomes. Additionally it is applied independently on each group of L bits, following the group crossover probability (P_{gc}). By always copying the successful parents to the next generation of solutions, as the cloning (elitism) operator suggests, we explicitly set the above probability to one for the successful and to zero for the unsuccessful parents. The crossover probability was chosen as $P_{gc} = 0.3$. The mutation probability was chosen as $P_m = 0.003$.

4.5.6. Reproducing new generations

The population size was kept small with 20 individuals per generation. The number of generations was not limited since the termination criterion will be reached after the completion of the user criteria.

4.6. Multi-objective genetic algorithm

The Evolutionary Declarative Design System is move towards the application of a different evaluation algorithm. Now on, design solutions are generated and evaluated automatically, without the designer intervention. The formation of such evolutionary environment is achieved through the creation of encapsulated design description, which defines a set of objects with relationships, as well as the generative principles involved during the generating process. Architectural design considerations are integrated into this process and made effective throughout in two ways, i.e., as knowledge and stylistic principles that can be built through the utilisation of object-properties-relations, and as specifications interactive supplied by designers as the selection and evaluation stylistic criteria. In order to approach the requirements of conceptual design we will concentrate on a multi-objective optimisation method such as a multi-objective genetic algorithm (). In this section the main focus is the development of a new generation engine is based on a *MOGA* within MultiCAD.

4.6.1. MOGA system

The algorithm developed is based on a Standard Genetic Algorithm [Goldberg 89]. Furthermore, specific mechanisms for representation and evaluation adopted. The theme is twofold.

- Description of both the genetic encoding of designs based on the representation adopted, and the associated genetic operators used in the evolutionary process.
- Definition of the methodology adopted for the multi-objective evaluation and selection of potential solutions.

The GA will evolve improved scene solutions by utilising multi-objective function.

4.6.1.1. Phenotype

The definition of scenes as phenotypes is based on the same simple geometry representation as introduced and explained earlier in current chapter. In order to have

better control over the morphology of the objects we separate the representation of a building into two geometric objects, one for spatial objects, and one for structural objects. In particular, it is used a quadrilateral that is capable to approximate curved surfaces. This quadrilateral continues to represent the spatial objects of a scene. The geometric object for the spatial objects requires 7 parameters for its geometric definition. The geometric object for the structural objects (roofs) requires 23 parameters. A phenotype of each design consists of a flat list of ordered parameters, which define the geometry of every basic object.

4.6.1.2. Genotype

The genetic operators of the GA are actually modifying the encoded scenes. The later are in the form of code scripts represented in a computer program as strings. Each code script represents a potential design solution, and composes one of the chromosomes in a population of GA the genotype. In the proposed system, every chromosome is arranged in a hierarchy consisting of multiple pieces of genes, for two fundamental reasons. Firstly, because of the application of declarative modelling by hierarchical decomposition. Secondly, because this arrangement is corresponding to the decomposition – part-of representation of a scene as it is normalised, by the *DKABM*. Finally each gene is being defined by float values. The length of a chromosome varies because different number/types of objects can be selected. However it is possible the generation of building designs with varying number of spatial-structural objects. In general, this chromosome may be mapped to a potential solution to the problem being investigated or as the design solution being developed in the case of our research. Based on the hierarchical design representation (Chapter 2), the phenotype is represented as the combination of spatial–structural objects and their associated properties. Since every object is defined by its associated object attribute parameters, each chromosome consists of a list of multiples of attribute parameters. The chromosome encoding scheme is based on the real lists, which is proved to be more suitable for this problem solving case. The proposed GA manipulates only the spatial and structural representation that is encoded in two chromosomes respectively in the genotype. Every spatial-structural object is defined by its associated feature attribute parameters. The GA will manipulate the values of these parameters, which they must be coded as the genotype. The attributes of a spatial or structural object include geometric features as well as other characteristic

attributes of this object, which as defined in the *MultiCAD Knowledge Base*. The chromosome scheme is distinguished between the two fundamental types of objects. The two types are: *Spatial* and *Structural chromosome*. The former represent spaces and rooms while the later represent structural elements, (roof, column). In this research the evolution of designs happen in two steps. During the first step it is used the *Spatial* chromosome, whiled during the second step it is used the *Structural* chromosome representing roof(s).

4.6.1.2.1. Spatial chromosome

The *spatial* chromosomes stand for the spatial objects of a declarative scene description. The geometric representation is coded as a quadrilateral isothetic object.

| Chromosome | |
|------------------|--------------------|
| Spatial object n | <i>Space index</i> |
| | <i>X</i> |
| | <i>Y</i> |
| | <i>Z</i> |
| | <i>Length</i> |
| | <i>Width</i> |
| | <i>Height</i> |

Table 4-3 Spatial Chromosome

A spatial chromosome is build up from 7 *genes*, (Table 4-3) the first is decoded into the type of space index, the next triple of genes correspond with positional three-dimensional coordinates of the lower left vertex of the quadrilateral. The next 3 numbers are decoded into the *Length*, *Width* and *Height* of a quadrilateral.

4.6.1.2.2. Structural chromosome

The structural chromosomes stand for the structural objects of a declarative scene description. We consider as a structural object only a *Roof*. According the geometric description of a *Roof* the chromosome has 28 *genes*. In particular, we consider that a Roof chromosome has the typical three genes for position, and three genes about length, width and height. In our case we decide that the attributes of positioning and length-width are defined from the space that a roof covers. The remaining genes that

characterise a *Roof* are D_{OFFSET_Length} , D_{OFFSET_Width} , O_{OFFSET_Length} , and O_{OFFSET_Width} , W_1-W_8 , and B_1-B_4 . The genes correspond to the parameters that control the forms of the object are showed in the following table (Table 4-4).

| Chromosome | |
|----------------------|----------------------|
| Roof object n | <i>Roof index</i> |
| | <i>X</i> |
| | <i>Y</i> |
| | <i>Z</i> |
| | <i>Length</i> |
| | <i>Width</i> |
| | <i>Height</i> |
| | D_{OFFSET_Length} |
| | D_{OFFSET_Width} |
| | O_{OFFSET_Length} |
| | O_{OFFSET_Width} |
| | W_1 |
| | W_2 |
| | W_3 |
| | W_4 |
| | W_5 |
| | W_6 |
| | W_7 |
| | W_8 |
| | B_1 |
| B_2 | |
| B_3 | |
| B_4 | |

Table 4-4 Roof Chromosome

4.6.1.3. Genetic Algorithm

The GA used within this stage is slightly different compared to the simple GA of the first attempt. In particular, the present GA includes the use of multi-objective evaluation techniques within the GA. It has the following outline. The GA begins with the creation of the first generation of solutions initialised with random values to allow

the evolution of designs from scratch. The evaluation of each individual solution during the generation happens with the call of relevant evaluation objective function(s). As a result, every individual solution gains a fitness value. The GA must then apply a ranking method for the individuals by calculating their single fitness values, and apply the selection operator in the form of a roulette wheel. The GA prefers individuals with higher fitness when picking 'parents' from the generation's population. The new population is generated with choice of parent solutions (with fitter solutions preferentially selected), and the application of the crossover and mutation operators. The next step involves the evaluation of the new genotypes and the same process is continued as before. This iterative process continues until either a specified number of generations (i.e. loops) have passed, or until an acceptable solution has emerged.

4.6.1.4. Evaluation functions

The objective functions of the system will be specific to the present building design application. The GA obtains information from the evaluation of the degree of performance for the individuals of every generation. In particular, a fitness function incorporates any possible design specification. It is obvious that it is highly problem-dependent, and in general is different for every design task. As soon as the evolutionary design environment deals with building design problems many of the fitness functions are the same and they could use for different building design problems, for example a fitness that concerns the area of a building, or its overall volume. Over time a large base of such fitness functions could be developed.

The applied objective function utilise either the information from the genotype directly and any other derivative performance data form it. This is happen as following: from a given scene description the user has already inputs a number of required-desired parameter set of values. At the same time from the structure of a scene representation the fitness mechanisms calculates some derivative data, which characterise the specific scene description in the form of properties, attributes. As a result every fitness function may obtain all needed information from a genotype and then it will evaluate it with its fitness values. For the adaptation to an architectural style the GA, use a number of separate fitness values as generated by the evaluation functions for each design, in order to guide evolution towards both 'improved' design

solutions. Here by the word improved we mean adapted to a particular architectural style.

4.6.1.4.1. Multi-objective functions

The problem of determination of the overall relative fitness of the genotypes it has its origins on the multi-objective character of design problems. In order to judging the overall fitness of the solutions we need an appropriate evaluation mechanism. From reviewing the literature in chapter two, we conclude towards the utilisation of a mechanism provided by Bentley [Bentley 97]. Such mechanism confronts two problems, the range of every objective and the importance of each weight. Therefore we imply a range-independent multi-objective ranking technique. Such technique enable an equally treatment of the objectives. Moreover it will avoid any laborious fine-tuning of weights. Following this, that technique will facilitate the specification of importance of each objective. Architectural styles have set of objectives that require different importance weight value.

The mechanism works as following. This aggregation based mechanism converts the fitness values for each objective into ratios, using the globally best and worst values. These global rations are then summed to provide an overall fitness value for each solution [Bentley 97].

Once the individuals in the population of the GA have been evaluated, the multi-objective method *Sum of Weighted Global Ratios (SWGR)* must calculate the overall fitness ranking position of each individual. First, *SWGR* records global minimum and maximum fitness values for each of the separate fitness values in each individual. For example, if the selected stylistic evaluation objectives calculate five separate fitness values for each individual, then *SWGR* holds a list of five corresponding minimum and maximum fitness values. These values are updated every generation by examining the fitness values of the new individuals in the internal population. If any fitness value falls below the minimum recorded, then this minimum value is updated. If any value falls above the maximum, then this maximum value is updated. These continuously updated minimum and maximum values, produced so far by a fitness function, give a steadily improving approximation of the effective range of that function. *SWGR* uses this information to convert every fitness value of every individual into a fitness ratio:

$$Fitness_ratio_i = \frac{(Fitness_value_i - \min(Fitness_value_i))}{\max(Fitness_value) - \min(Fitness_value)}$$

Such approach allows the equal treatment of all evaluation criteria. On the other hand it eliminates the difficulty of having multiple criteria with different effective ranges by equalising all of the effective ranges. Consequently, the individual's ratios are summed, and each fitness value multiplied by its relative importance weight. In this way a single overall fitness for each individual is achieved.

4.6.1.5. Genetic operators - Crossover

The genetic operators are transformed only genotypes, i.e. coded designs. In this second stage of experiments this genotype differs from the earlier application in MultiCAD-GA. The genotype consists of a hierarchically decomposed chromosome. Multiple blocks of seven genes compose the chromosome. We follow this arrangement because of two reasons. Firstly Declarative modelling by hierarchical decomposition, and secondly, the structure of the *DKABM* framework of the knowledge representation used to define the phenotypes, with each block of genes being a coded primitive shape and each gene being a coded parameter. However it is possible the generation of building designs with varying number of spatial-structural objects. In the current *MOGA* we have used the single point crossover method. The single point crossover practice defines that in every step of the evolution process a crossover point is being randomly defined. The crossover point defines the chromosomes to be passed into the child atom by the two parents. The chromosomes before the crossover point derive from the first parent and the chromosomes after the crossover point from the second parent..

4.6.1.6. Mutation

We apply two different types of mutations. The first is *Random* mutation. In this case for each variable that is going to be mutated, choose a random value within its range and assign this value to the variable. Therefore every value is possible. We implement the *Random* mutation which helps the *MOGA* explore a greater space on the problem space. The mutation occurs at the end of every evolutionary step except the final one, so as not to mutate the resulting population. The mutation procedure defines a random number in the range in which the variable to be mutated is defined. The second

method is the *Exponential* mutation. This mutation type comes from the idea that the role of mutation at the beginning is to make large jumps whereas later on, as the search progresses it should be used more for fine-tuning so small jumps are more desirable. After some initial experiments we decide to use *Random* mutation.

4.6.1.7. Selection

The selection technique that selected is based on fitness proportionate selection. In particular we imply the *Stochastic Universal Sampling* scheme [Mitchel 89]. In order to preserve individuals corresponding to good solutions we also implement the elitism scheme.

4.6.1.8. Initialisation

The genotype consists of the string of integers, upon which the reproduction operators of GA manipulate, and the phenotype contains the string real variables. Before the initialisation, the range of integer and the upper and lower bounds of each real variable, which represents a corresponding attribute parameter, are read from an initialisation file which contains the user requirements in the form of the scene description. During the initialisation, each gene of the genotype in GA is randomly generated as an integer, within the range. All fitness values for each member of the population are also initialised (to zero).

4.6.2. Implementation of Evolutionary Design Environment

We imply an evolutionary design environment by combining the MultiCAD system architecture and a genetic engine. A *MOGA* is a generative engine built in MultiCAD, which generates solutions following design goals using heuristic algorithms. As the designs evolved solutions approach a level of improvement accepted by the designer. The *MOGA* refines the values of the design parameters represented in the scene description as they introduced by the designer. Given a scene description, there exist input parameters and output parameters. The *MOGA* considers the constraints as environment and the parameters as genes. Further it search for optimal values for the parameters in order to satisfy the design constraints and finally to achieve a design goal. As the different constraints appeared the genetic algorithm could guide the search towards their direction.

The whole procedure happens according to the following steps. A designer introduces a scene description declaratively. At the same time a number of different types of relations appeared between sub-scenes- part of scenes and their parameters. In the next step the designer introduce some preferred goals that a scene-sub-scene/design-sub-design should achieve. In particular a goal define one or more objective function and chosen values. In order to define an objective function specific parameters from the scene-design are obtained. In our system we utilize a number of specific elementary parameters in order to construct the gene representation. It is possible to define some second-degree parameters in order to obtain better control over the formation of complex fitness functions. Following the hierarchical decomposition of a scene and the character of the *DKABM* framework a set of constraints is emerged among scene-design parameters. Depending on the design goals the designer has the possibility to have a detailed control over these parameters during the design process. In our current application we consider such a constraint set as pre-defined within the GA. On an alternative approach the architect has the opportunity to search for a design applying some constraints already existed within the knowledge base of MultiCAD. Such a base could be the *Style base*, which includes relative knowledge in the form of rules and constraints about specific vernacular architectural styles. Such a constraint's regrouping could help the designer easily apply the same set of parameter for the design of a different building artefact.

According the MultiCAD system architecture, we have already defined a standard representation of scenes, in our case buildings. In particular we imply standard pre-defined parameters as these stored in the respective bases of MultiCAD system. In addition we imply specific pre-defined *inner* and *inter* –relationships between building parts, (*spatial* elements, and *structural* elements). Through a *DKABM* the designer could define any element of a building. For example, the spatial element room has the following parameters (*Length, Width, Height, Position, et cetera*).

Chapter 5

Evolutionary Declarative Design Prototype System

5.1. Introduction

In this chapter we will present the implemented system components that are crucial for the testing our initial hypothesis. Therefore we will interfere with specific sections of the system framework of MultiCAD as it was presented in Chapter 2. The current chapter is divided in three sections. In the first section we present an initial attempt to imply MultiCAD system architecture within a commercial Computer-Aided-Design (CAD) and database software.

In the second section we present a first attempt for the introduction of a genetic algorithm within MultiCAD software.

The third section is the significant section of the chapter. We present in details the full implementation of the proposed evolutionary declarative design system. In particular we present significant changes within the total MultiCAD system architecture. The main advantages were implied during the description and generation phase of the declarative conceptual cycle of MultiCAD. A new version of MultiCAD system has developed in collaboration with the TEI of Athens. The latest version based on the ‘*dot.Net*’ programming environment. In parallel, a different commercial CAD software is utilised for the visualisation phase, that of VectorDraw.

5.2. DKABM mechanism implementation

Here is briefly presented a first implementation. The next and current implementation is analytically described in a following section of the chapter. We will provide an application framework where we show in which way the introduction of architectural knowledge and in particular architectural style affects the description phase and the impact of architectural style in the description of a scene. The adoption of *Declarative Knowledge for Architecture-oriented Building Modelling (DKABM)* produces

coherent architectural building descriptions. In advance in a higher degree of exploration of architectural knowledge we expressed models that incorporate sets of particular constraints such as architectural style and building typology principles [Ravani 03]. The software application is based on a commercial CAD system in order to support the declarative process of developing coherent building models based on architectural knowledge. A main contribution of this work is a new way to describe declarative buildings models through dialogs or Graphical User Interfaces, based on architectural style knowledge. MultiCAD's evolution can manage the elements of the *DKABM* and offers the possibility of using *Normalized Declarative Building Model (NDBM)*. The information provided by this framework is stored in the *Knowledge Base* along with the architectural constraints. We have developed several mechanisms that check the consistency of any building description introduced to MultiCAD II and correct it to respect the restrictions of *DKABM*. We have also developed a graphic user interface to confront the architectural knowledge. That interface completes the MultiCAD II information system in order to handle the data of the *Knowledge* and the *Scene Base*. Moreover, the user can introduce the *DKABM*'s knowledge in MultiCAD II and manage the *Normalized Declarative Building models* of every scene. We applied the MultiCAD software architecture in a general-purpose commercial computer aided design system, (AutoCAD 2002). For the reason that MultiCAD II is a multi-layered architecture, for each layer, we developed a series of components that enables the linkage within AutoCAD (Figure 5-1).

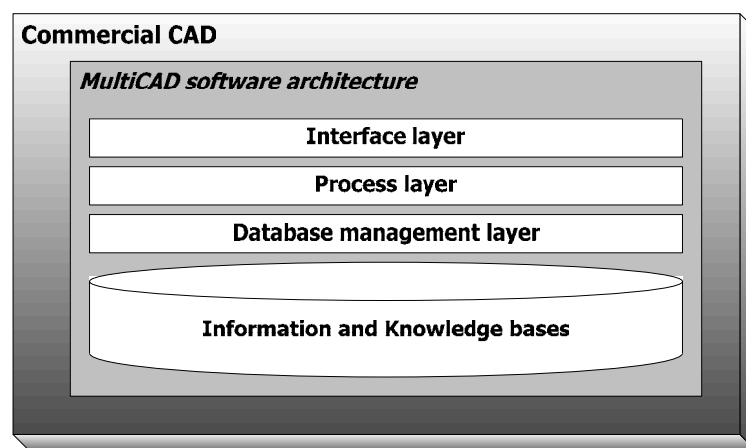


Figure 5-1 MultiCAD II Software architecture in a commercial CAD system.

We implanted the components to AutoCAD software in order to take advantage of its functionalities for description, geometric modelling and visualisation of scene models.

5.2.1. Components of DKBM implementation

The sets of developed components appear in three layers. Interface, Process and Database management layer are interpenetrating within the three phases of the declarative conception cycle. These layers are implemented with the use of Visual Basic for Applications supported by AutoCAD.

5.2.1.1. Interface layer

The components of Interface layer support the phase of description and the scene understanding on the declarative conception cycle.

Description phase: For the description phase we generate user interfaces for normalised declarative scene description. We define a new way of declarative description of *NDBMs* based on dialogue, and knowledge-driven interface components. These interfaces enable both parametric and declarative description of scenes and they support constraints as drawn from the applied *DKABM*. Furthermore, the designer can use stylistic and building typology knowledge.

Scene understating phase: For the Scene understanding layer a series of interfaces were designed in order to visualise the generated scenes in a user-friendly manner. The visualisation was made within the design environment of AutoCAD. In this way, we take full advantage of advanced CAD functionalities provided by a system such as AutoCAD. These functionalities are among others great variety of file formats, model editing, et cetera.

5.2.1.2. Process layer

The components of the Process layer support the first two phases of the declarative conception cycle, the description and generation phase.

Three distinct components are embedded during the *Description phase*:

- A component for logical error-handling. It supports the designer with error warnings. These errors prevent descriptions from containing contradictory constraints such as for example *Building Area* cannot be greater than *Site Area*.
- A component that ensure that input description satisfies the constraints drawn from the applied *DKABM*. The input description is corrected to match up with the decomposition hierarchy, the permissive and obligatory constraints of the *DKABM* by generating respective objects, relations and properties, if needed.

- A component that controls the way that architectural constraints of style and building typology affect the input description. The control follows the *DKABM* correction process, but now focuses on more specialized constraints.

During the *Generation phase* one component controls the generation of valid scene solutions that verify the input description's constraints (designer constraints, applied *DKABM* constraints). The scene solutions are modelled geometrically through the comprehensive geometric modeller of AutoCAD design environment.

5.2.1.3. Database Management layer

For data management we developed components that interact with MultiCAD's database. In particular data such as architectural knowledge and building scene models could interact with the *Knowledge* and *Scene Base*, of MultiCAD II.

Description phase: For the description phase two components are developed. The first component interacts with Multi-CAD's *Knowledge Base*, where the architectural knowledge is already stored. The second component manipulates the Scene Base of MultiCAD in order to store and handle the normalised declarative scene descriptions (*NDBMs*).

Generation phase: for the generation phase it is developed a component that stores the scene solutions derived during the generation phase. Such storage happens within MultiCAD's *Scene Base*. Furthermore, solutions are stored in a form that implies mainly their geometric information.

5.2.2. Using DKBM in Design cases

We apply examples from three design cases to check the effectiveness of the proposed application, two habitations and an office building. Each of the two habitations belongs in different architectural style, Santorini style, and Metsovo style. Both regional architectural styles introduce a specific typology of spaces, and their shape is expressed through distinct characteristic geometrical primitive forms. Both styles were presented in details in chapter 3 and 4. The third design example belongs to a typical office building.

5.2.2.1. Interfaces

The components developed in order to apply MultiCAD software architecture in AutoCAD are underlain behind a series of interfaces that help the designer to follow

the declarative conception cycle and in parallel utilise sophisticated CAD functionalities in a user-friendly manner. The main interface concerns the parameters of the normalized scene description. The input process imply both declarative and parametric manner. Figure 5-2(a) shows the main interface, and illustrates an example of user's input description for a building according to a style, that of Santorini. The main sections of this interface concern alternatives (e.g. different architectural styles-building types) and scene description parameters. The interface, also, provides the designer with a selection between two different types of description approaches: parametric, (exact numeric values) and declarative (vague expression of design intentions). In parallel with the designer's input process, the components of coherency are activated to support the description phase:

Conflicted parameters are controlled.

Accordingly to the type or style of building chosen, some of the properties as appeared on the interface are locked to default values. For example, in Santorini style, roofs of buildings are combination of vaulted and flat form, by default.

The normalized scene description is ensured, by generating the respective entities, properties and relations according to the applied *DKABM*. Figure 5-2 (b) shows the decomposition tree of the normalised scene description for a building of Santorini style.

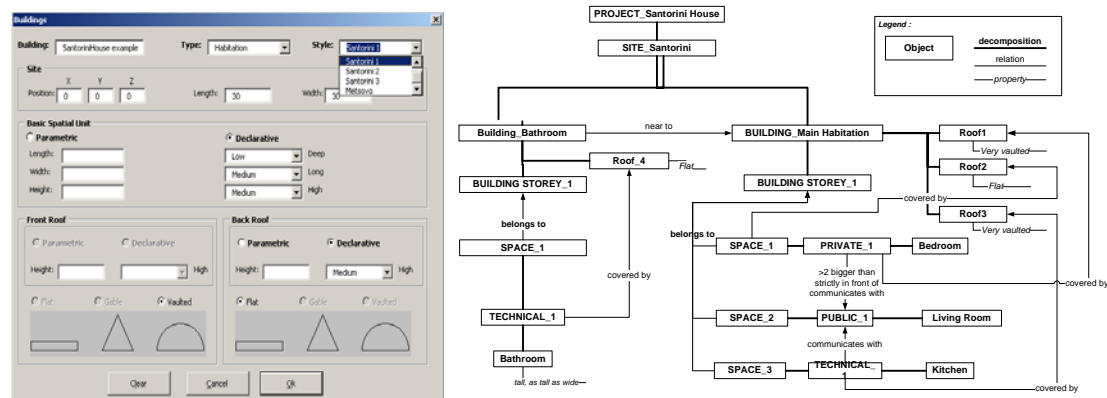


Figure 5-2 a) Interface for the description of normalized scenes, b) An example of normalized description of Santorini's style home.

The next phase concerns the generation of valid scene solutions. Solutions, (building) are modelled geometrically by AutoCAD's solid modeller, and they are visualised

through its design environment. In Figure 5-3a is illustrated a valid solution that satisfies the input description of a *Santorini* style building.

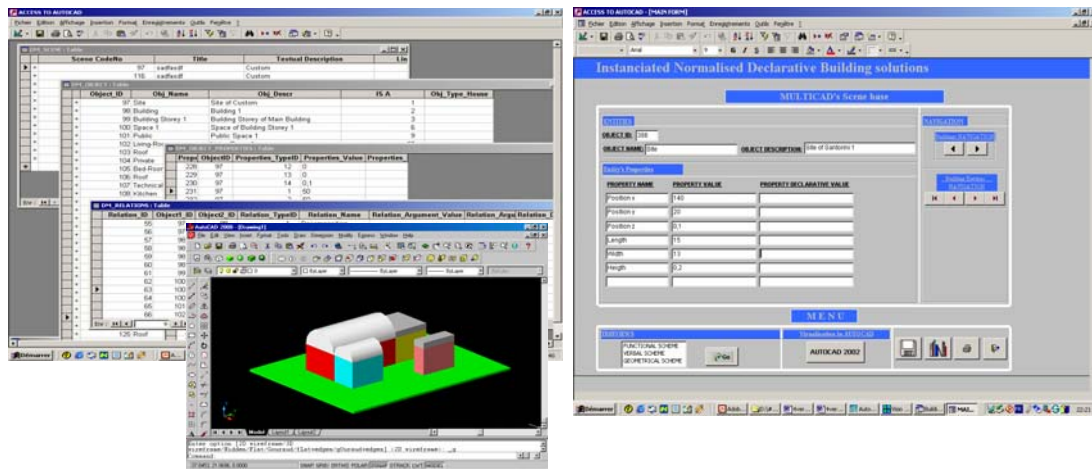


Figure 5-3 a) A scene solution of Santorini house and the respective Scene Base records. b) Interface for the data management of scenes.

In order to manage the *Scene Base* of MultiCAD, we have developed a series of interfaces handling the respective records of the scene solutions and provide further functions of editing and manipulation of the later. The developed database management components provide the designer with various functionalities. For example, a main interface of data management is filled with the records of the entity *Site* of the above *Santorini* building solution (Figure 5-3(b)).



Figure 5-4 Interface of tree view representation of scene solutions.

In order to facilitate the phase of scene understanding, the building solutions can be represented in the form of three different tree-views. In particular, they incorporate information concerning decomposing entities of the building focusing on the function of spaces; verbal description of the building's decomposing entities and geometric primitives of every space entity. For example, in Figure 5-4, the solution of *Santorini* building (shown in Figure 5-3) is represented respectively in the form of the three tree-views. In order to take full advantage of the database management components and in specific the editing tools that they offer, records (building solutions) stored in the *Scene Base* may be fed back to AutoCAD in order to be modelled and visualised in its design environment.

In this way, the designer has the ability without inputting any scene description (in fact, bypassing the description and the generation phase), to choose and edit any description of building solutions already stored in the *Scene Base* and visualise them in AutoCAD for eventual further manipulation. Supplementary interfaces provide information about the building entities in a data-sheet view. They offer the means to the user to both edit a parameter of building entities and then visualise the respective building to AutoCAD, or directly visualise building solutions without further changes. We also input the description of a building of Metsovo style and that of an office building. Figure 5-5 illustrates an example of obtained valid scene solutions (the colours represent different type of spaces).

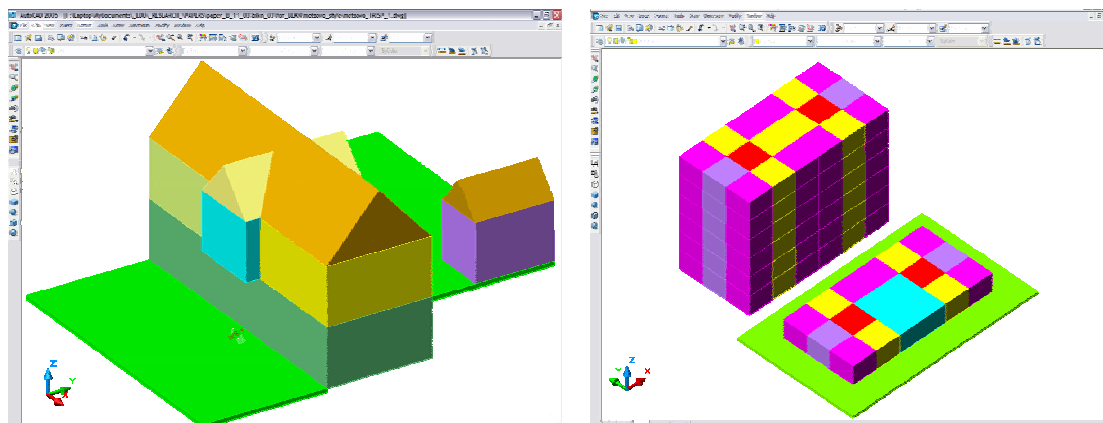


Figure 5-5 Examples of scene solutions

5.2.3. Discussion

In this first implementation it is developed a software application based on a commercial CAD system in order to support the declarative process of developing coherent building models based on architectural knowledge. A main contribution of this work is a new way to describe declarative buildings models through dialogs or Graphical User Interface, based on Architectural knowledge. Moreover we propose a layered architecture-driven development of MultiCAD type software applications, based on a general-purpose commercial CAD and database systems. In specific, the implementation is based on AutoCAD 2002 and Access 2000, and provides a test-bed for the above approach. The resulted dialogue-based description of declarative models offers many improvements. In particular:

- Facilitates the capture of user requirements in a declarative way during the early phase of design process
- Permits the editing of design parameters and constraints in any phase of the declarative process in an interactive and user-friendly way
- Favours the completeness of normalised building scenes
- Facilitates the development of interface components in order to support this kind of description
- Provides coherency mechanisms according to the applied DKABM, which are enriched with specialised constraints concerning architectural style and building typology.

The proposed layered architecture-driven development of MultiCAD applications takes advantage of the existing components of commercial CAD and database systems. It also encourages the development of professional quality and easy extensible MultiCAD applications with rich capabilities of data integration and exchanges. The applications examples show, in three design cases, the easy build of user-friendly design environments for early phase of architectural design process while proving the effectiveness of the proposed schema. The current implementation provides further development of *domain-specific* environments based on declarative knowledge. Moreover we improve the architecture-driven development of MultiCAD applications and the procedure of the dialogue or graphical user interface generation in a semi-automatic manner.

5.3. MultiCAD – GA

This section presents the organisation of the generic evolutionary design system and gives an overview of the elements of the system. Each of these four elements are fully explained and justified.

5.3.1. Basic implementation structure

The MultiCAD-GA genetic algorithm-based engine is described in this section. In the beginning we argue for the choice of Genetic Algorithm (GA), as a fruitful generation engine. Then we present the system in details. Specifically, we present, how the terminal subscene bounding boxes of the internal tree model are encoded into chromosomes, which genetic operators are used and in what way and, finally, how new generations of solutions are reproduced and evaluated by the user.

GAs are used as a feasible generation engine mechanism within MultiCAD system. The GA is the basis for the proposed declarative design generative process, as shown in Figure 5-6. The successful formation of a GA program needs the mutual use of appropriate data structures that correspond to the chromosome representation, and appropriate algorithms that correspond to “genetic” operators that transform one or more individual chromosomes.

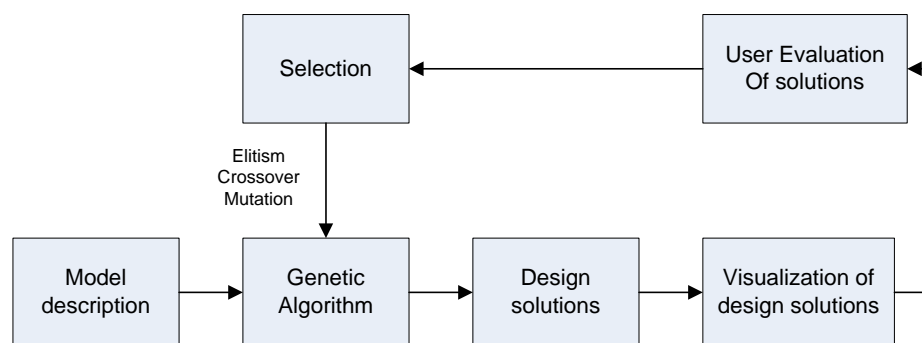


Figure 5-6 MultiCAD-GA system organisation

In our system we make use of an improved GA, which is based on the SGA [Goldberg 89]. The genetic operators, the crossover and mutation adopted in this approach are similar to that of SGA, but with several improvements based on the consideration of the special needs of architectural conceptual design. The modification concerns the initialisation of the system for the generation of first generation. The genotypes will

have the form-scheme of hierarchical decomposition. The phenotypes will be defined by a 3d mesh geometry representation with low control parameters. The GA will evolve improved solutions by utilising a single objective function. Figure 5-7 illustrates how the GA allows the evolution of a range of different designs.

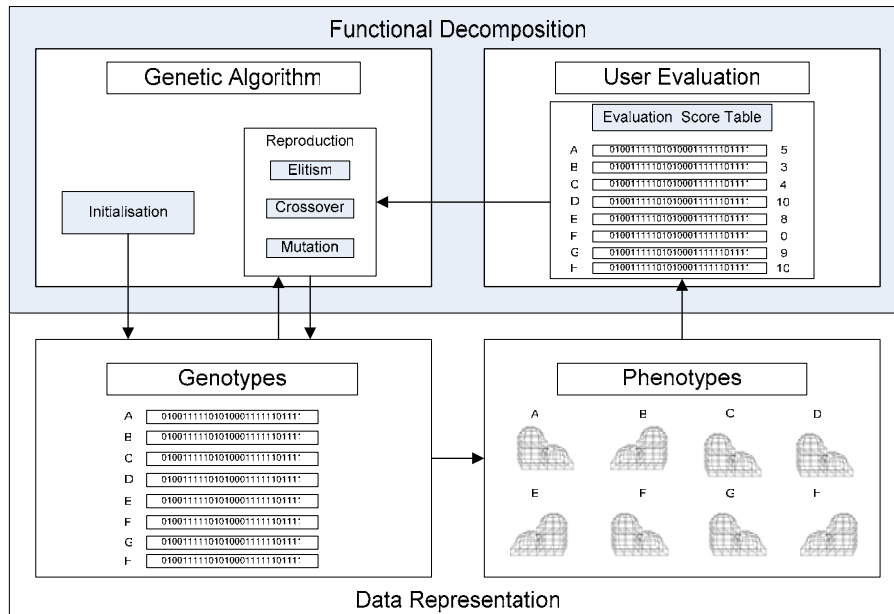


Figure 5-7 MultiCAD-GA Functions & Data structures

5.3.2. Implementation and simulations

In MultiCAD-GA it is used the main user interface of MultiCAD providing a way to introduce the scene description. As it concerns the generation engine there were no specifically designed graphic user interface. The input process is all text based, and consists of two main steps. Several simulations have been performed using the MultiCAD-GA solution search engine for different scene descriptions. Each solution was then visualized through a VRML file, which allowed the user to project it onto any viewer (Internet Explorer). The time required for a complete production of the next generation of 20 chromosome solutions varies from a few seconds to a few minutes depending on the severity of the problem constraints and on the choice of the crossover and mutation probabilities. Moreover, another advantage of the genetic search engine over the traditional exhaustive linear search strategy is that the user not only directs the search on more promising parts of the search tree but also examines and evaluates the scenes in groups of 20 rather than one by one as in MultiCAD-II. The latter allows for a relative comparison and is found to generally improve the

subjective quality of the final solutions. Finally, Figure 5-8 shows two solutions for a residence scene, generated with MultiCAD-GA. These solutions were derived within the first ten generations.

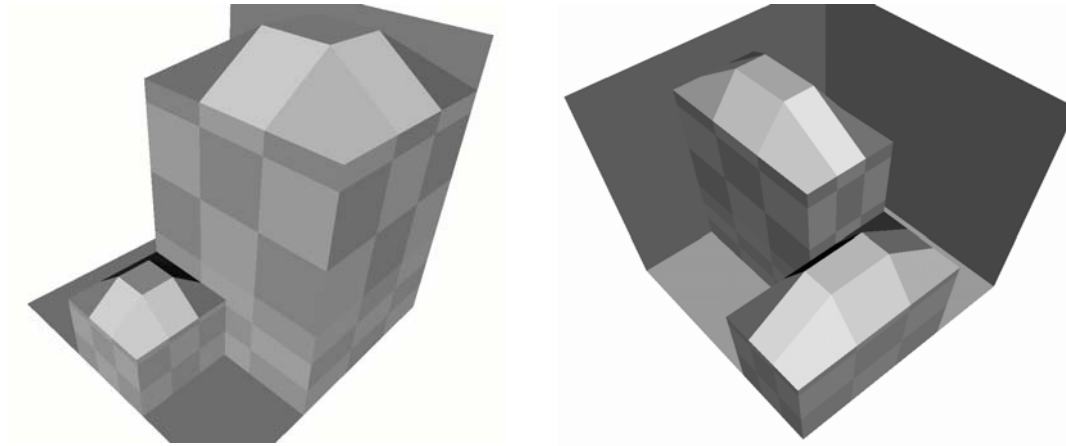


Figure 5-8 Two solutions for the residence scene

The results obtained through simulations are quite encouraging and show that MultiCAD-GA can be a useful tool in applications involving 3D form design. The main problem was how, and at what level, to incorporate declarative description's constraints into the GA. The IDR input format used by MultiCAD-MultiForm provided a practical representation for describing building geometry, but presented no formal way to incorporate information about relations between the variables under study, which was one of the mechanisms intended for the designers to describe and control design intentions. In the current version of the software it is within the GA code that the constraints encoding architectural design intentions are implemented.

In the next section the MultiCAD- system is oriented towards the following directions. In terms of input, a graphical user interface (GUI) is created to help designers insert necessary information, which would automatically upgrade the input files and the GA code. We imply a CAD interface that provides building-related information in a 3D model. A crucial factor is to be able to use standard graphics formats, as the ones generally used by architects in their practices. MultiCAD-GA will be able to generate a new, modified 3D model of the building, based on the rich information achieved by the MultiCAD system, to allow for quick visualization of results and further development and changes from the designer.

5.4. MultiCAD – MOGA

In this section of the chapter we present the implementation of the proposed Evolutionary Declarative design MultiCAD system.

In a dedicated declarative design system at the beginning of the design process, the design problem is defined by an abstract declarative design concept without having to specify many detailed parameters to consider. The process continues with the generation via the evolutionary process of more specific and desired designs that are developed from this declarative concept. A declarative evolutionary design process consists of three cyclically linked stages - the “*Design Proposal Description*” (Description phase), the “*Design Generative-Developmental*” (Generation phase) and the “*Scene-Solution Understanding*” phase, (Figure 5-9).

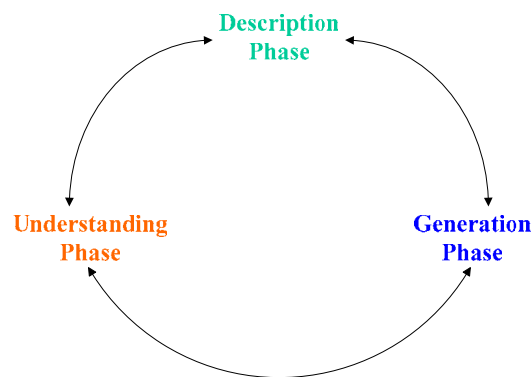


Figure 5-9 Declarative evolutionary design process

5.4.1. Functional aspects of methodology

We provide a flowchart in order to show the general flow of methodology (Figure 5-10). The methodology follows the Declarative Evolutionary Design Cycle and also provides the sequence of the phases as implemented. The model description phase is broken down to the declarative *Model description* and the *Style Selection*. Both of these activities can be user inputted or make use of stored data. The Generation phase is supported by the generators parameters definition and the solutions generation as well as the generation engine (multi-objective genetic algorithm) fine tuning which is also the feedback supplied by the solution understanding phase.

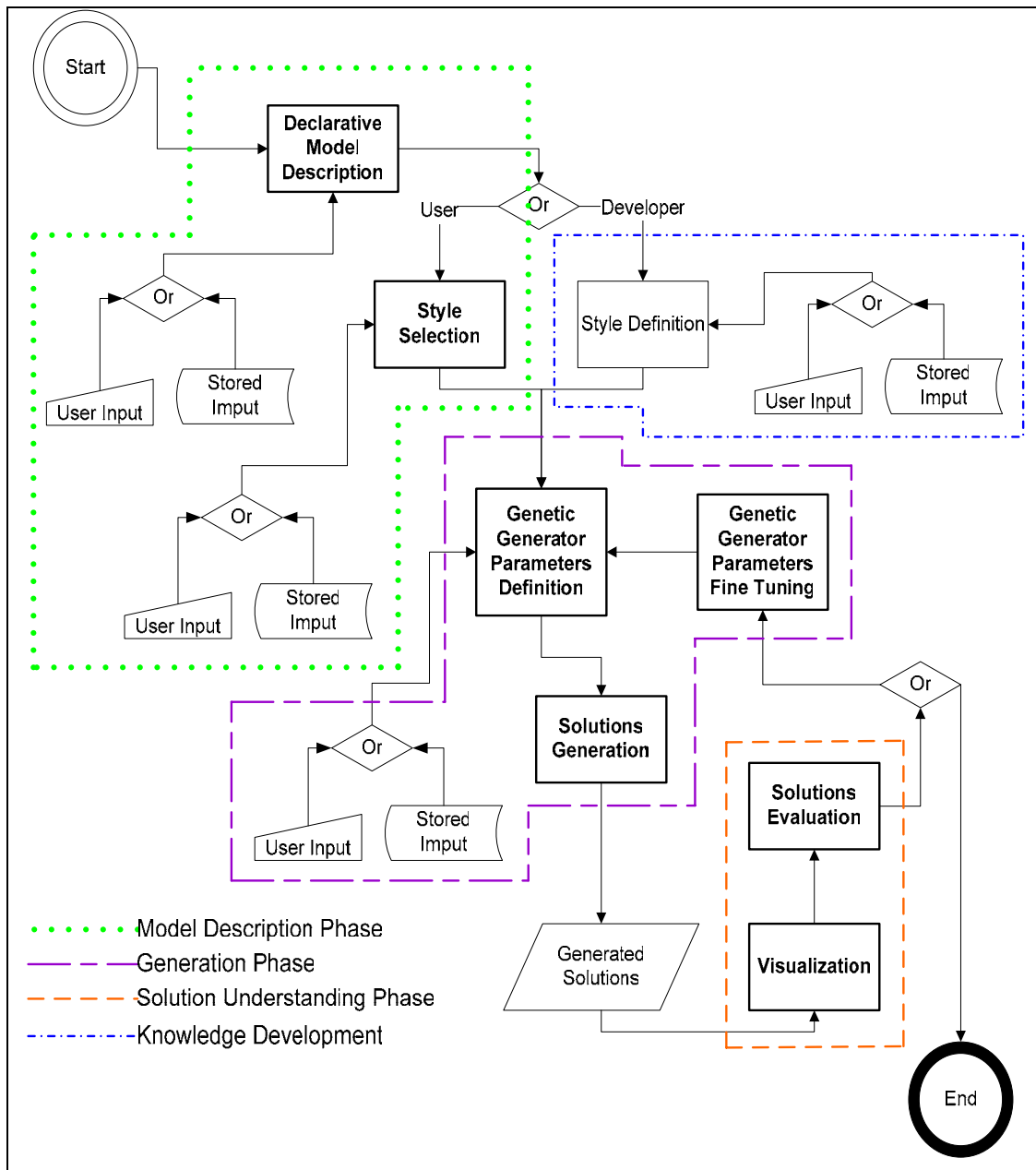


Figure 5-10 Flowchart of applied methodology

The generation phase leads to the generated solutions which are data stored in the Scene base of the MultiCAD *Knowledge base*. The Solutions understanding phase consists of the graphical visualization of the generated solutions and the evaluation of the user that can lead to further fine tuning and solutions generation or to the end of the flowchart. The knowledge development is the phase where the *developer* (expert) defines both the attributes of *DKABM*, and architectural styles for use in the style selection by the *designer*.

5.4.2. Software architecture

The architecture of the current implementation is based on the architecture proposed by the MultiCAD framework. Therefore, the software is divided into three main layers that have specific roles in the implementation. The three layers are the following: the *User Interface Layer*, the *Processing Layer* (the solutions generator) and finally the *Data Management Layer* (Figure 5-11).

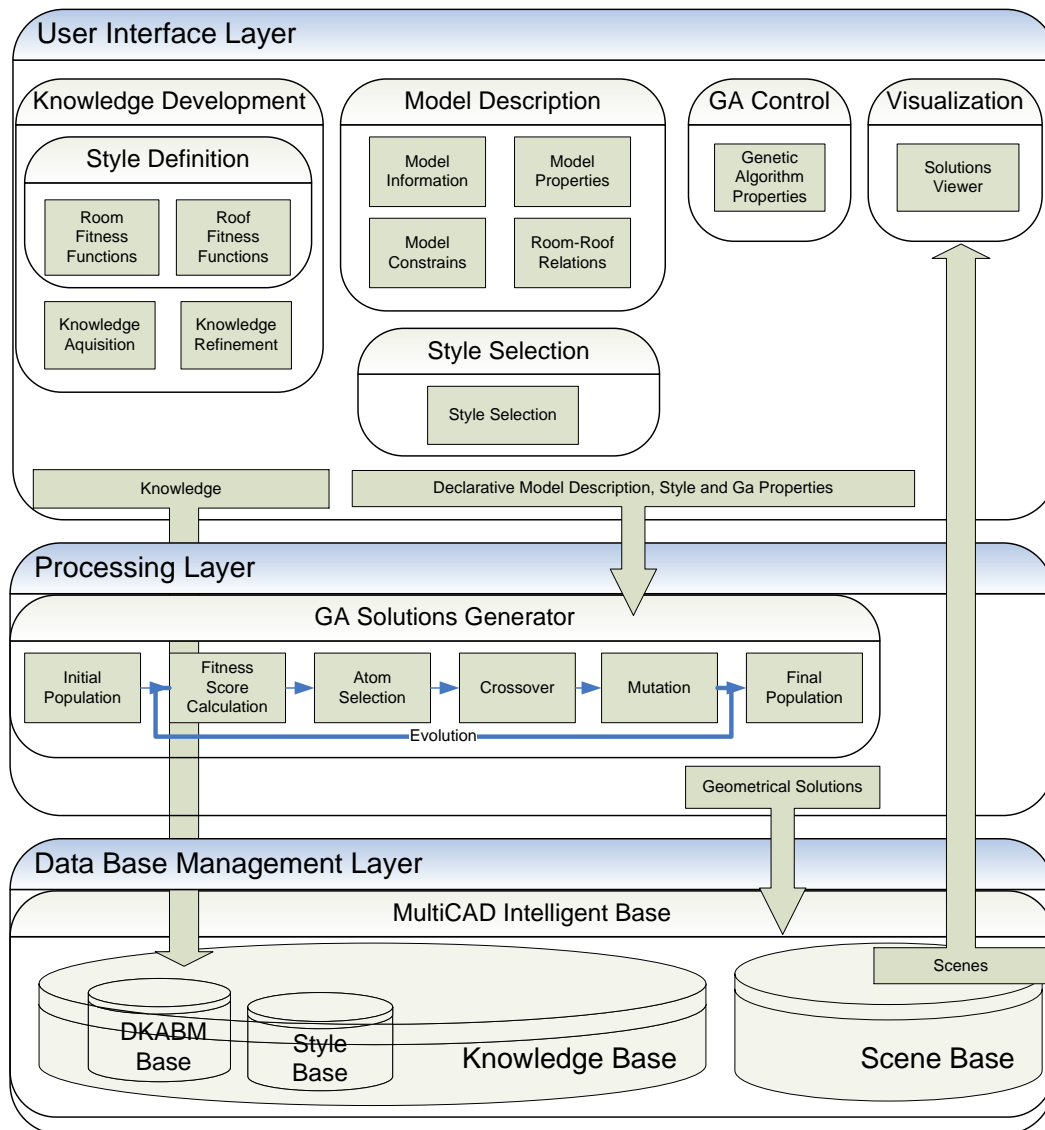


Figure 5-11 MultiCAD MOGA Implementation Architecture

The *User Interface Layer* is the topmost layer and consists of the front end interfaces with which the user interacts. The *Processing Layer* is the middle layer and contains the solutions generator which receives information from the top layer and creates

geometrical solutions. The bottom layer, the *Data Management Layer*, concerns the data and knowledge storing and retrieving. Besides it is the layer that receives the geometrical solutions from the processing layer and knowledge from the topmost layer via knowledge acquisition.

5.4.2.1. User Interface Layer

The *User Interface Layer* consists of all the forms and interfaces that the user has access to. Moreover it is divided into the following five main components (Figure 5-11).

- *Declarative Model Description component.* The set of forms with which the user presents the model to the system. Here the user defines model information such as number of rooms, room types, site's properties, rooms' properties, relations between rooms and roofs and hard constrains.
- *Style Definition component.* It consists of two forms that are accessed only by the developer (power user) and give him/her the ability to define fitness functions and weights, for rooms and roofs respectively, which will define an architectural style.
- *Style Selection component.* This is the equivalent of the Style Definition component for the simple user. In this case the user will be called to select a style among a list of predefined, by the developer, architectural styles.
- *GA Control.* It is the interface where the GA's parameters are defined. They are the number of atoms per population, the number of generations, the probability of mutation, the atoms' max age, the elitism et cetera, and have great influence over the outcome of the generator. Moreover in the GA control component the user defines parameters concerning the number of repetition (loops), the number of atoms per generation to be stored in the *Scene Base*, the number of generations required to pause the evolution in order to have the ability to visualize partial results.
- *Visualization component.* It is a component that accesses the stored geometrical solutions and presents them to the user graphically.
- *Knowledge Development component.* The knowledge development component is the interface that provides the system with knowledge concerning the DKABM and architectural styles.

5.4.2.2. Processing Layer

The *Processing Layer* consists of one key component that is the *Genetic Algorithm Geometrical Solutions Generator*. This component receives the declarative model description, the style information and the GA properties previously defined in the upper layer and computes geometrical solutions genetically (Figure 5-11). For the computation of the geometrical solutions, the *GA Geometrical Solutions Generator* component follows five main steps.

- *Initial Population*. The solutions generator receives the declarative model description and based on that, builds the initial GA population. The initial population is setup based on the properties defined in the user interface layer and the model constrains.
- *Fitness Score Calculation*. Once the initial population is built the GA computes the fitness scores of the current population based on the fitness functions defined in the user interface layer.
- *Atoms Selection*. The individuals' selection receives the calculated fitness scores and selects individuals accordingly to fill up the matting pool for the evolution of the population.
- *Crossover*. In this stage individuals from the mating pool are selected and combined to build up the evolved atoms in the new population.
- *Mutation*. A possibility of mutation was defined in the previous layer in the GA control component and based on that some individuals of the population will be mutated.
- *Final Population*. The final population is the evolved set of generated solutions. This stage is reached only when the defined, in the GA properties, generations have finished. If there are more generations left then the evolution after the mutation stage will return to the fitness calculation stage for further evolving.

5.4.2.3. Developers Features

In the *Developer Feature*, the *developer* (when defined in the *GA Control* component) can make use of an extra feature provided for the calculation of fitness function weights. The *developer* (expert) defines the set of fitness functions in the previous layer that represent an architectural style. The evolution of the population begins and

stops when a predefined number of generations have passed (Figure 5-12). The system presents the population to the developer and waits for manual score input. The developer will evaluate the atoms and provide the scores. The system will then calculate the new weights of fitness function for the functions using the ‘*Generalized Inverse*’ technique. The new weights will reflect directly the developer’s preferences and the evolution will be affected accordingly. This feature is a direct knowledge refinement tool which can fine tune architectural styles according to the developer’s (expert) evaluations, who plays the role of the knowledge engineer.

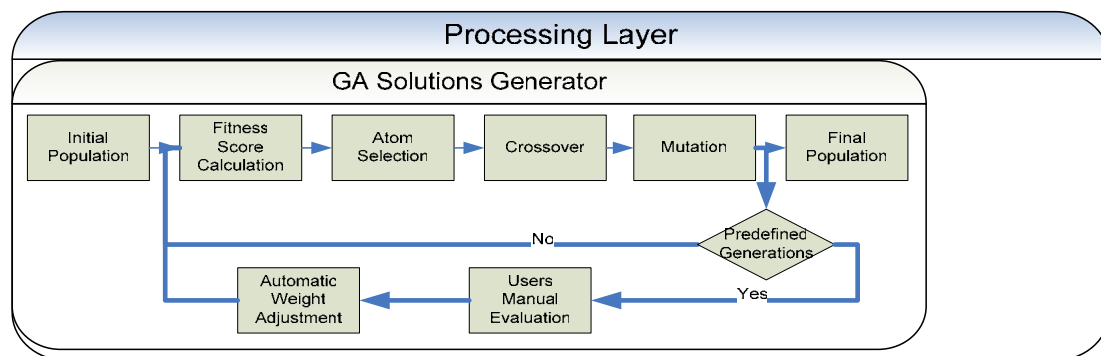


Figure 5-12 Manual score input for automatic weight calculation

5.4.2.4. Data Management Layer

The final layer is the *Data Management Layer* which contains the MultiCAD intelligent Base. The intelligent database consists of two main data bases which communicate with each other (Figure 5-12).

These bases are the *Knowledge base* and the *Scene base* (Figure 5-11).

- *Scene Base*. The *Scene base* consists of multiple data tables which store all the information concerning the scenes, solutions, fitness functions’ scores and weights along with generations and repeat times (Figure 5-13).
- *Knowledge Base*. The *Knowledge base* contains two bases: the *Style base* and the *DKABM base*. The *Style base* contains all the information concerning architectural styles. The *DKABM base* holds the information concerning DKABM patterns.

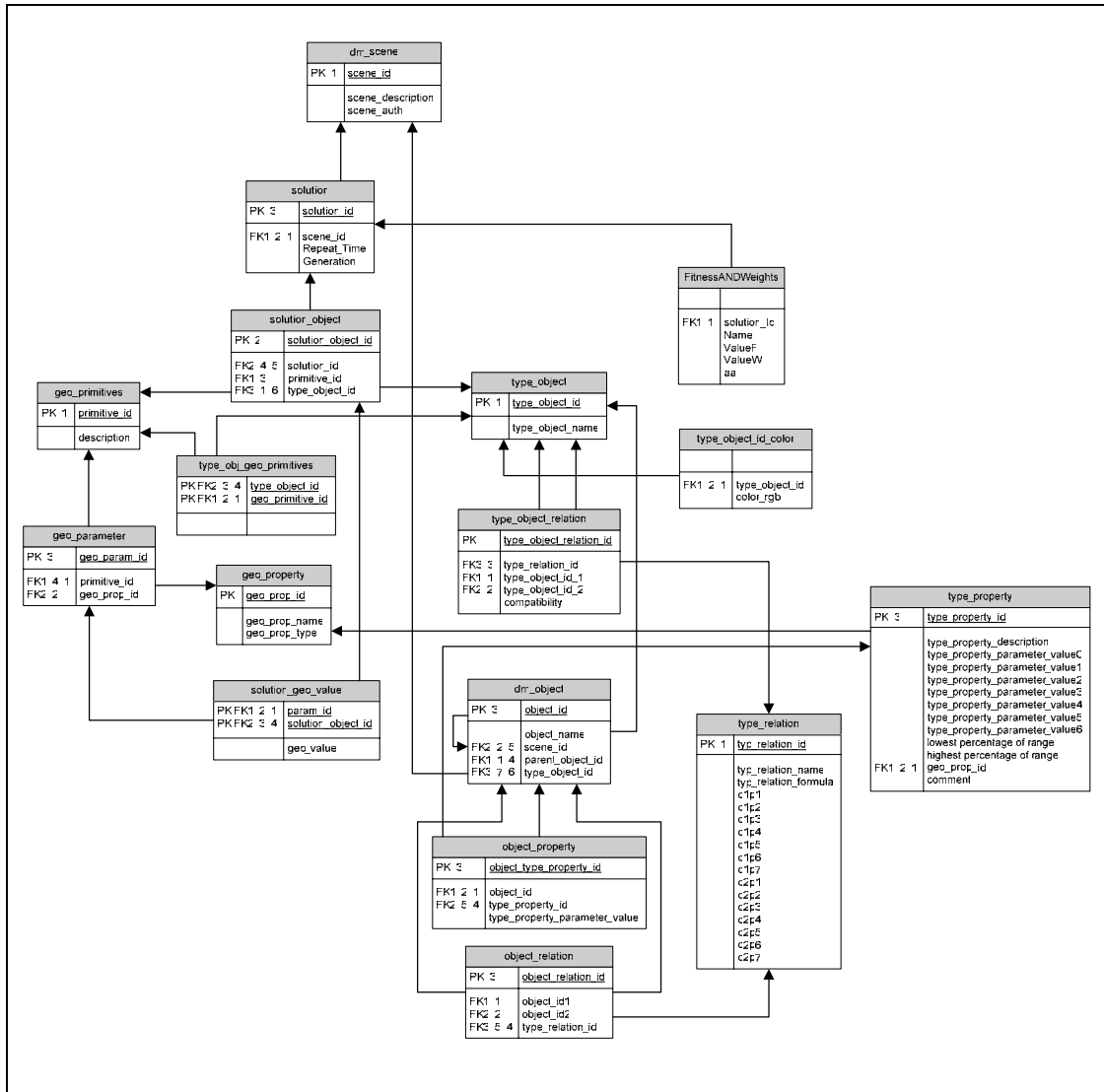


Figure 5-13 Scene base

5.4.3. Functional models

The IDEF [IDEF 95] diagram (Figure 5-14) shows the implementation within the declarative cycle as divided in four main activities. Those activities are the *Description Phase* (Activity A1), the *Generation Phase* (Activity A2), the *Visualization – Understanding Phase* (Activity A3) and finally the *Knowledge Base Management – Development* (Activity 4).

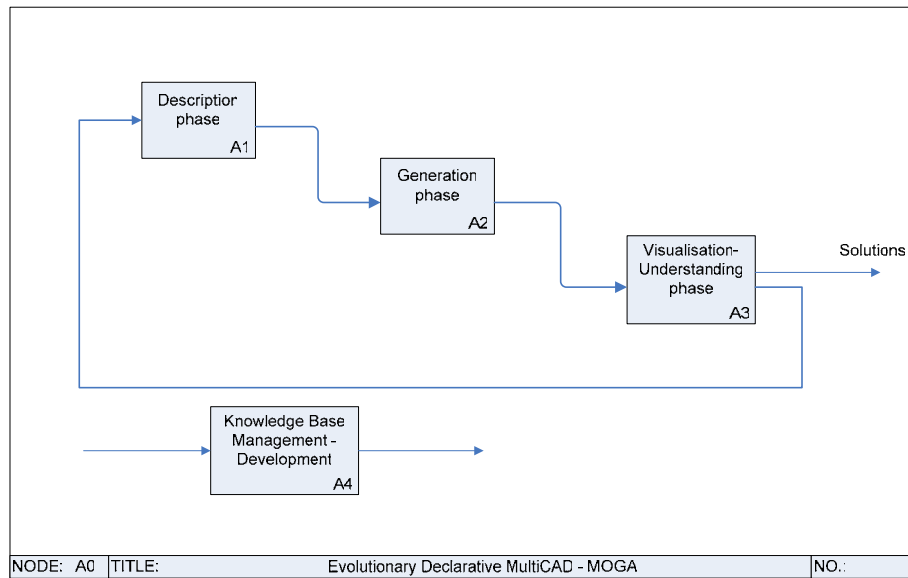


Figure 5-14 Idef Level 0

5.4.3.1. Description phase – Model Description (Activity A1)

The model used in this implementation is complex enough to hold the information needed. As a result the description of the model used by the GA must be derived from multiple tables created by the use of user-friendly interfaces. The starting interface is the ‘*Model Information*’ dialog which supplies the *developer / user* with information regarding the room types, the number of roofs and the site’s properties (Figure 5-15 Activity A1.1).

With the use of the previous information a general table is created and presented to the *developer / designer* in the next dialog which is the ‘*Model Properties*’ dialog. In this dialog the *developer / user* can define more subtle properties of the model such as the ranges where rooms could be placed within the site, as long as their height, width and length range (Figure 5-15 Activity A1.2). From the ‘*Model Properties*’ dialog a ‘*Rooms*’ table is created which will be later used in the following procedure of GA. The next dialog concerning the model description is the “*Constrains*” dialog where the user is called to choose constrains amongst the previously defined rooms. From the Constrains dialog a ‘*Constrains Table*’ is created and stored for later use in the GA (Figure 5-15 Activity A1.3). The last dialog concerning the model description is the Room-Roof Relations dialog where a ‘*Roof X covers Room Y*’ type table is created (Figure 5-15 Activity A1.4).

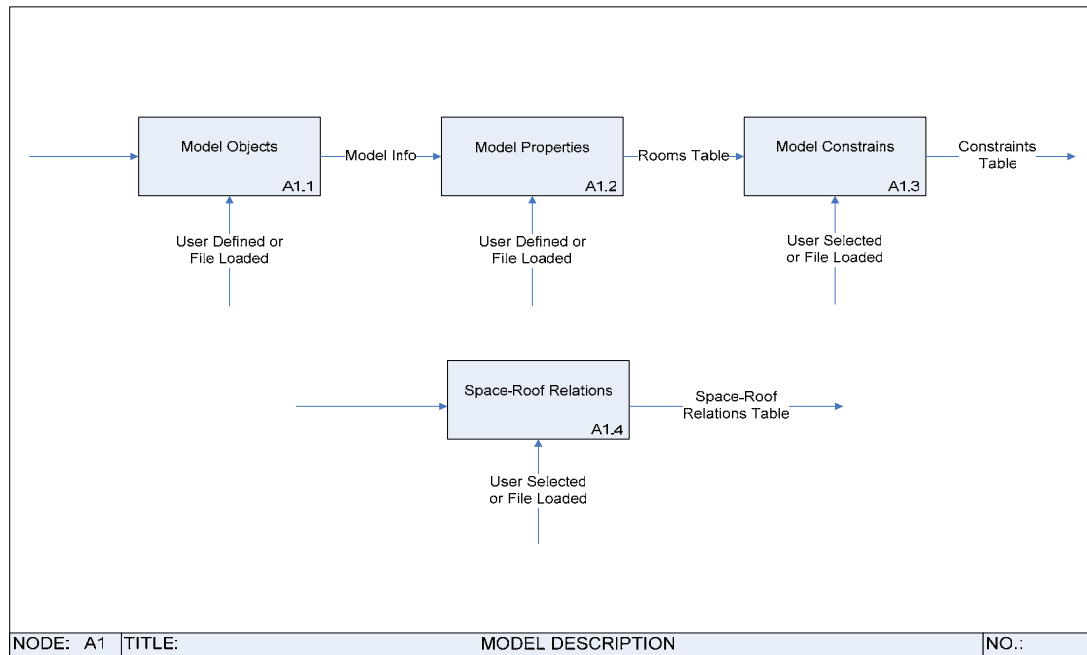


Figure 5-15 Model Description Activity

5.4.3.2. Description phase – Fitness Function Selection (Activity A1)

Furthermore the model description encloses the selection of fitness functions which is broken down into two activities: the selection of room and roof fitness function respectively or the Style selection (Figure 5-14 Activities A1). Both activities are supported by the appropriate interfaces. In particular, in the ‘*Room Fitness Functions Selection*’ dialog, as well as in the ‘*Roof Fitness Functions Selection*’ dialog, the *developer* selects fitness functions for rooms or for the whole site and for the roofs (Figure 5-1 Activity A1.5 – Activity 1.6). In the background a ‘*Fitness Function*’ table is formed, which contains all relative objective fitness.

Finally, from those two dialogs, two tables concerning fitness functions are created, the first for the rooms and the second for the roofs. In the *Style selection* dialog the *designer* is called to choose from a set of already defined architectural styles (Figure 5-16 Activity A1.7).

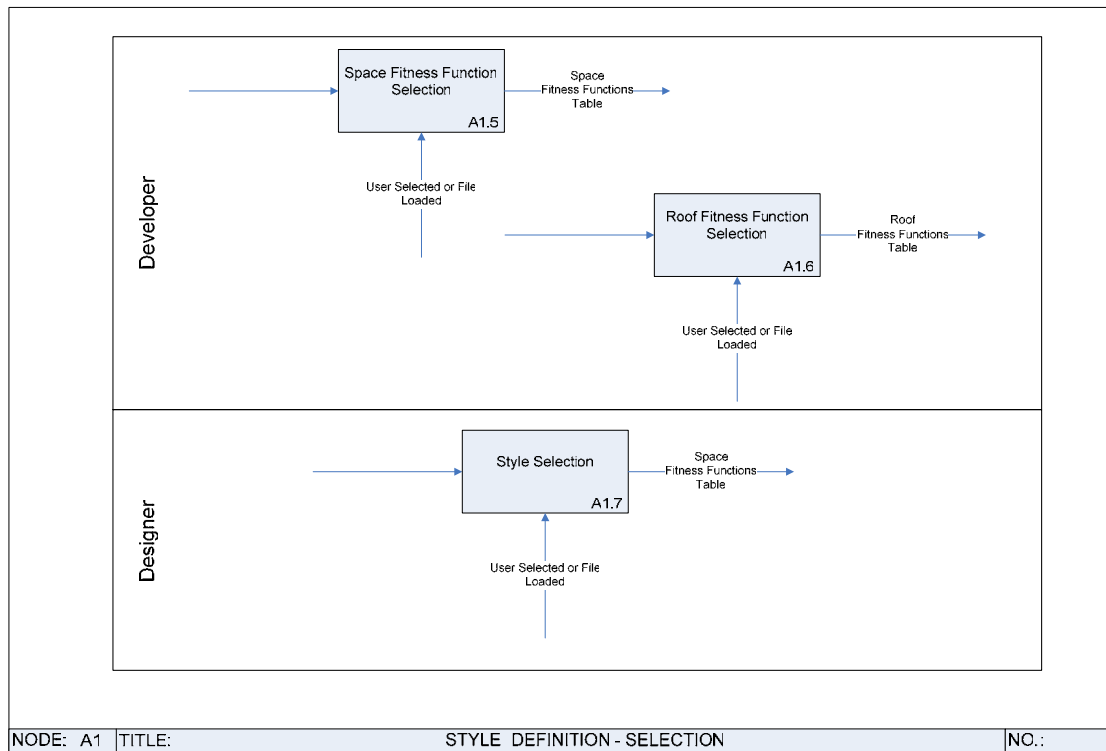


Figure 5-16 Fitness Function Selection Activity

5.4.3.3. Solutions Generation (Activity A2)

The solutions generation is accomplished in two steps. The first concerns the *spatial composition* of the rooms and the second the *morphological appearance* of the roofs (Figure 5-17 Activity A2). Both activities require some common genetic algorithm properties which are supplied by the appropriate interfaces supported by the GA console dialog for *Rooms* and *Roofs* accordingly.

For the generation of solutions we make use of the previously created tables. For the spatial modelling of the rooms we use the *Room*' table, the *Constraints* table as well as the *Rooms Fitness Functions* table (Figure 5-17 Activity A2.1).

For the morphology of the roofs we make use of the '*Room-Roof Relations Table*', the *Roofs Fitness Functions* table and the site chosen by the user (Figure 5-17 Activity A2.2). Both activities will store the outcome in the MultiCAD Scene base for visualisation and management.

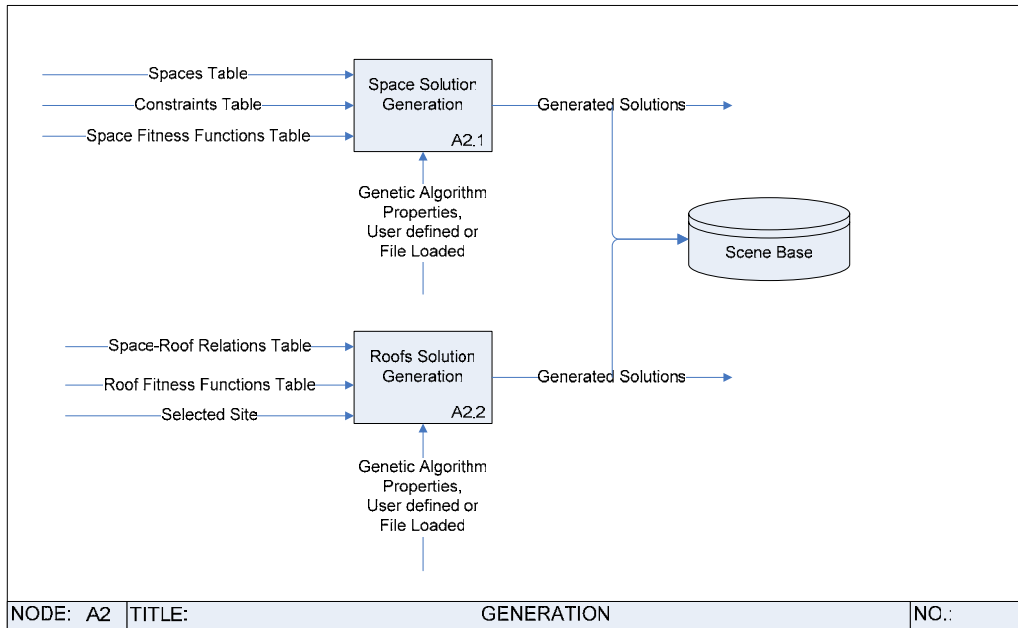


Figure 5-17 Solutions Generation Activity

5.4.3.4. Visualisation Understanding (Activity A3)

The Visualization - Understanding activity provides the graphical representation of the generated solutions. In the Solution Visualisation activity (Figure 5-18 Activity 3.1) the generated scene solutions are visualised.

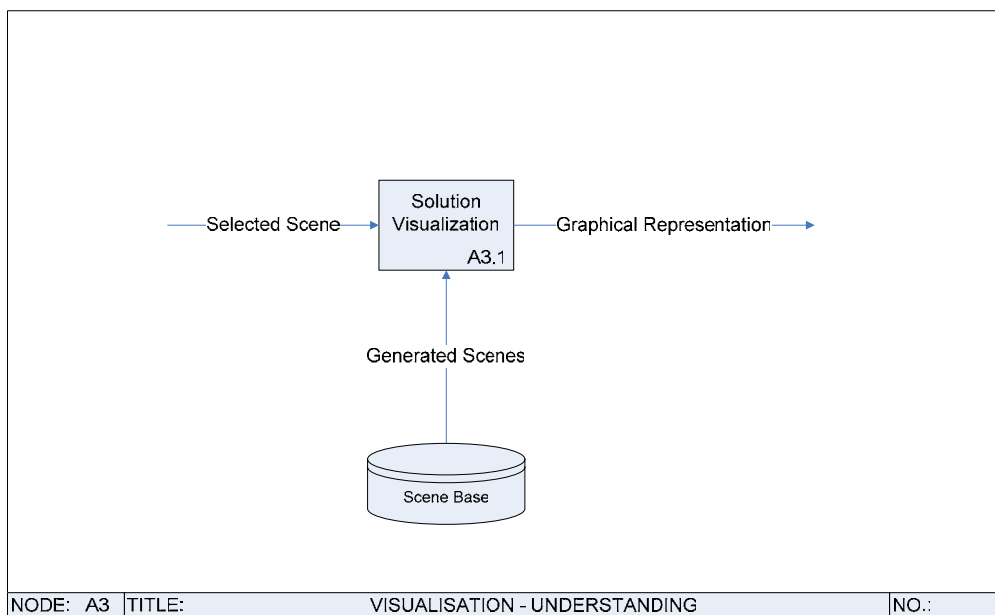


Figure 5-18 Solutions Visualisation Activity

5.4.4. Components Interface

The topmost Idef diagram shows that the implementation is divided in three main activities. Those activities are the Model description, the fitness functions selection and finally the solutions generation.

5.4.4.1. Model Description Interfaces (Activity A1).

The model used in this implementation is complex enough to hold the information needed. As a result the description of the model used by the GA must be derived from multiple tables created by the use of user-friendly interfaces. The starting interface is the *Model Information* dialog (Figure 5-19) which supplies us with information regarding the room types, the number of roofs and the site's properties (Figure 5-15 Activity A1.1).

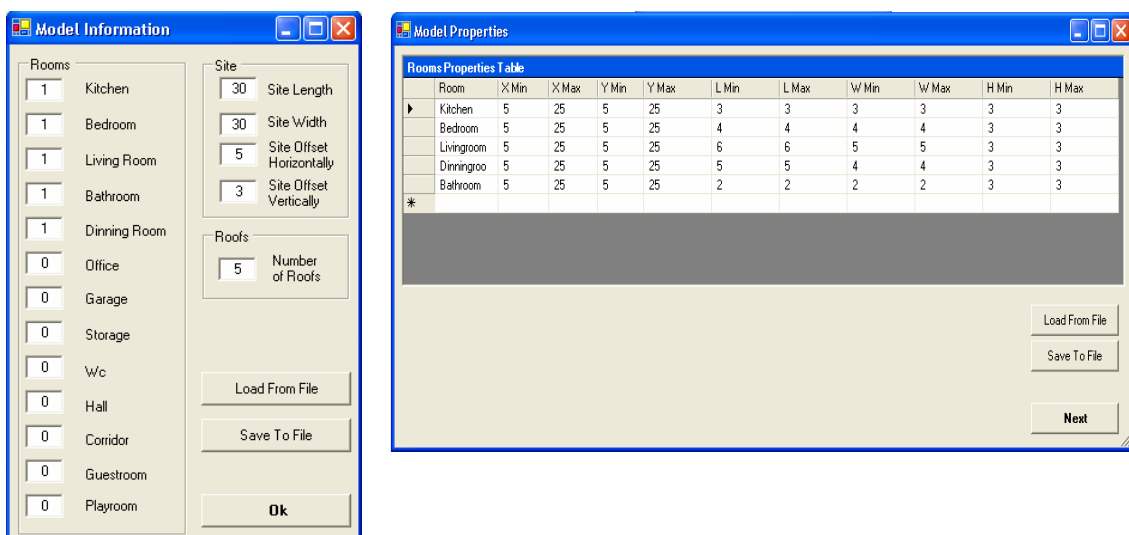


Figure 5-19 *Model Information & Model Properties* interface

With the use of the previous information a general table is created and presented to the user in the next dialog which is the *Model Properties* dialog (Figure 5-19). In this dialog the user can define more subtle properties of the model such as the ranges where rooms move as long as their height, width and length range (Figure 5-15 Activity A1.2). From the *Model Properties* dialog a *Rooms Table* is created which will be later used in the GA. (Figure 5-19).

The next dialog concerning the model description is the *Constrains* dialog where the user is called to choose constrains amongst the previously defined rooms (Figure 5-20). From the *Constrains* dialog a *Constrains Table* is created and stored for later

use in the GA (Figure 5-15 Activity A1.3). The last dialog that concerns the model description is the *Room-Roof Relations* dialog (Figure 5-20) where a *Roof X Covers Room Y* type table is created (Figure 5-15 Activity A1.4).

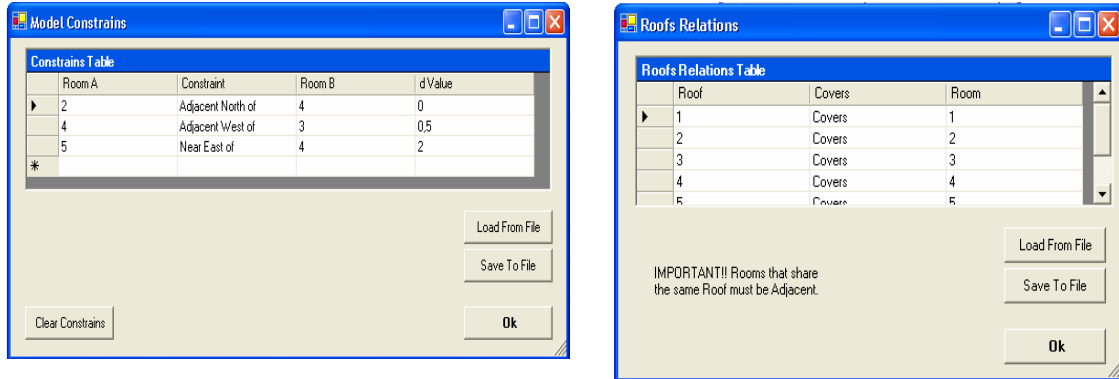


Figure 5-20 *Model Constraints & Roofs Relations* interface

5.4.4.2. Fitness Function Selection Interfaces (Activity A1).

The selection of fitness functions is broken down in two activities which are the room and roof fitness function selections, which are supported by the appropriate interfaces. In the *Room Fitness Functions Selection* dialog, as well as in the *Roof Fitness Functions Selection* dialog (Figure 5-21). The user selects fitness functions for rooms or for the whole site and in the background a *Fitness Function Table* is formed. From those two dialogs, two tables concerning fitness functions are created, the first for the rooms and the second for the roofs (Figure 5-16 Activities A1.5 and A1.6).

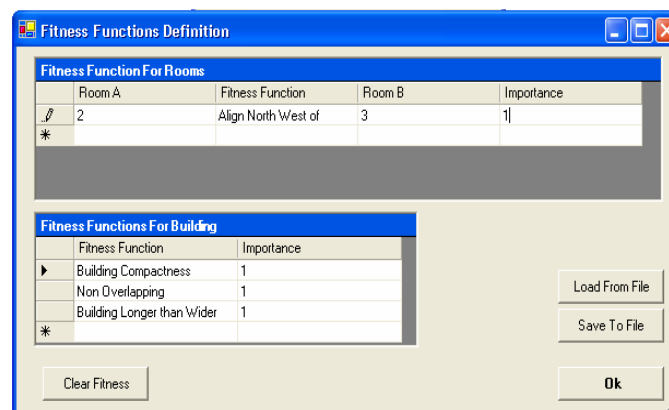


Figure 5-21 *Fitness Functions Definition* interface

5.4.4.3. Solutions Generation Interfaces (Activity A2).

The Solutions Generation is accomplished in two steps. The first concerns the spatial modelling of the rooms and the second the morphological appearance of the roofs.

Both activities require some common Genetic Algorithm properties which are supplied by the appropriate interfaces supported by the GA Console Dialog for Rooms and Roofs accordingly (Figure 5-22). Moreover in the GA control interface the designer defines parameters concerning the number of repetition (loops), the number of atoms per generation to be stored in the *Scene Base*, the number of generations required to pause the evolution in order to have the ability to visualise partial results.

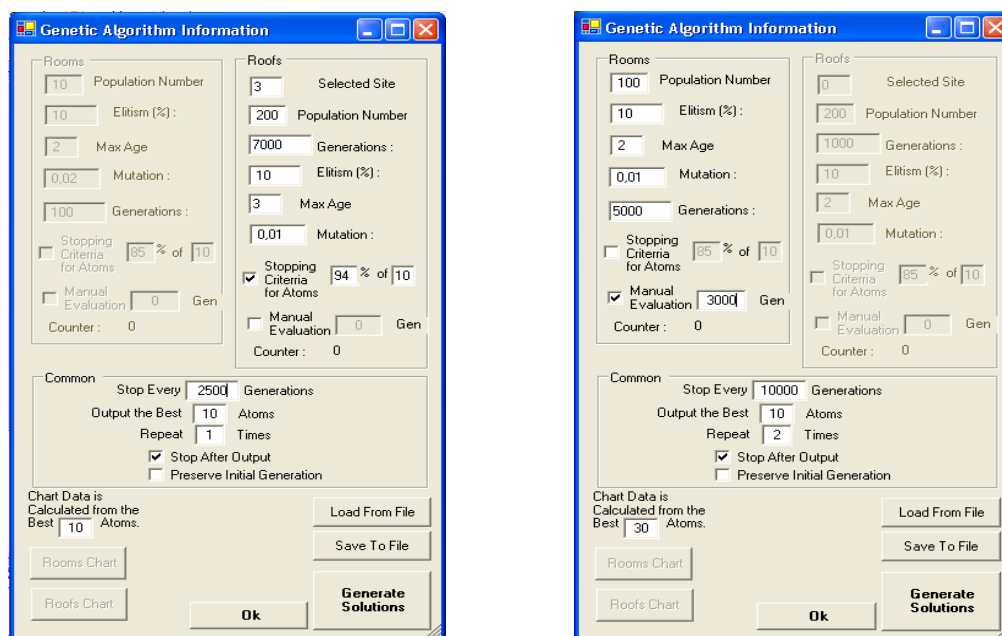


Figure 5-22 *Solutions Generation* interface

For the generation of solutions we make use of the previously created tables. For the spatial modelling of the rooms we use the *Rooms Table*, the *Constraints Table* as well as the *Rooms Fitness Functions Table* (Figure 5-17 Activity A2.1). For the morphology of the roofs we make use of the *Room Roof Relations Table*, the *Roofs Fitness Functions Table* and the site chosen by the user (Figure 5-17 Activity A2.2).

Both activities will store the outcome in the MultiCAD *Scene Base* for visualisation and management of the generated solutions.

5.4.4.4. Visualization-Understanding Interface

The Visualization interface consists of a graphical viewer based on the VectorDraw drawing component. The viewer is a powerful tool because it provides to the designer both geometrical and non-geometrical information related with the generated scene solutions (Figure 5-23). In particular, it is divided in three areas. Firstly, in the main area (centre) it is the drawing area that provides two and three dimensional aspects of the scenes. Secondly, on the left area the designer can view the fitness functions that participate in the evolution, their score and the weight factor of each of them. Thirdly in the right area the designer can view two type of information. On the upper side the designer have access in the number of repetitions of the GA and the number of generations for each repetition. In this way he/she can visualise any solution. On the lower side, the designer has access in the geometric data that characterise the viewing scene. Moreover he/she could alter any of the model's parameters.

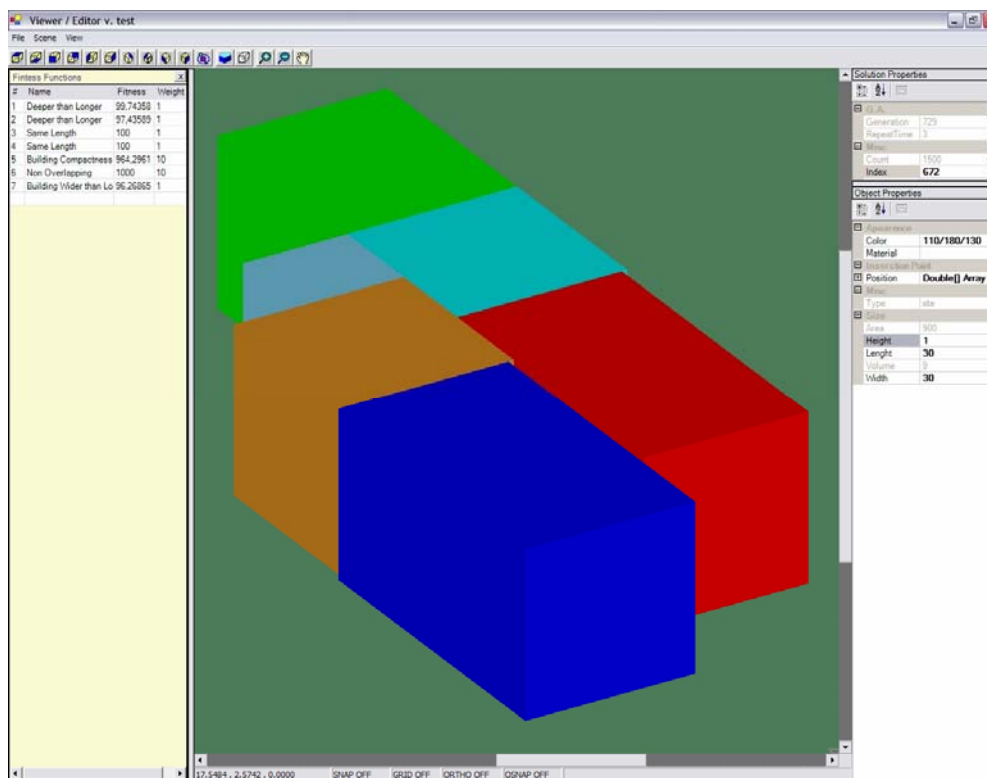


Figure 5-23 Graphic visualisation interface

5.5. Discussion

The improvement of both the input and output formats of the initial evolutionary declarative environment would mean more than simply making it design-oriented and thus more prone to be used in architectural design. The implication of an appropriate computer graphics paradigm interface, it provides to Evolutionary Declarative design MultiCAD system a potential for making this declarative design system able to confront diverse class of design problems not explored until today.

The implementation of Evolutionary Declarative design methodology with the incorporation of architectural knowledge illustrates the potential of MultiCAD-MOGA for the conceptual architectural design with the use of architectural style. The integration of MultiCAD software architecture with computer aided design tools provide architects with an influential infrastructure for architectural design.

Chapter 6

Experiments – System Evaluation

6.1. Introduction

In this chapter we perform a series of experiments for the evaluation and validation of the evolutionary declarative design system prototype. The experiments are based on the application of style in architectural design. In particular, first we imply the *Santorini* style and second the *Metsovo* style. In this way the plan of the experiment is organised as following. In the first stage the evolutionary process concerns adaptation of a building to stylistic criteria of spatial planning. In the second stage the evolutionary process concerns adaptation of the roof(s) of a building to stylistic criteria of roof morphology. Therefore the chapter has two sections one for each style. Each section has two subsections, one for evolution of space planning and one for roof morphology.

6.2. Santorini Style

In this experiment we try to evaluate the performance of the evolutionary design system when it is confronting building designs with no more than six spaces and at the same time with few hard constraints among them. The spaces could have varied dimensions while they are only constrained by the building brief. Stylistic principles expressed as objective functions direct the evolution of the individuals in order for the later to express building designs adapted to the *Santorini* architectural style. The whole process has two distinct steps. Firstly, the designer utilise the MOGA system in order to evolve the spatial planning–spatial composition for given building requirements, under specific principles of *Santorini* style. Secondly, from the solutions of the earlier step, in particular from the resulted individuals (scenes) of the last generation, the designer selects one individual in order to evolve the roof morphology of the building following specific principles of *Santorini* style.

6.2.1. Space Planning

In this section of the current experiment we provide results concerning the evolution of building designs of given requirements, under specific compositional principles of the *Santorini* architectural style. The input process has the following steps along the MOGA system:

- Introduction of type and number of all spaces of the building. Additionally the designer defines the number of roofs for the building. Definition of the site dimensions and possible offset from the boundaries of the site.
- Introduction of information about the placement of all spaces as a range that is limited by the site limits, it is possible to select some fixed position for one or more spaces within a particular part of the site. Next, the designer defines the range of their length, width and height. It is possible to define either fixed and/or variable dimensions for each space.
- Definition of the specific requirements and demands for the building as constraints between the spaces. Such hard constraints will not get violated during the process of evolution. Such relations, in general, they have the form of binary topological relations.
- Selection of an appropriate style module that the designer wants to apply for the design evolution of the building. He/she follows specific steps in order to assign some of the spaces to the specific principles of the selected style, in this case the *Santorini* style.
- Implementation of the genetic algorithm with some pre-specified properties. The designer could redefine such parameters that control the implementation of the genetic algorithm in order to adapt better to a particular design problem.

6.2.1.1. Scene A

We define now the appropriate objects of *Scene A*. The object properties and some relations are considered as hard constraints. We consider the stylistic criteria as soft constraints which are the evaluation criteria.

6.2.1.1.1. Definition of Scene

In this step it is defined the *Number of Rooms* and the *Types of Rooms*. In parallel it is provided information of the *Site* dimensions. Finally it is introduced only the number

of roofs that the model might have. In this case study a building configuration is introduced that has totally six spaces and three roofs. The dimensions of the Site are 25 by 25 units, with an offset from each side by 5 units. All units are in meters. The quantification of the site is set in steps of 0.10 meters. The definition of the *Number of Roofs* is made now, but this information will be used during the step of roof morphology. In Figure 6-1 is appeared the form with relevant data for the model information.

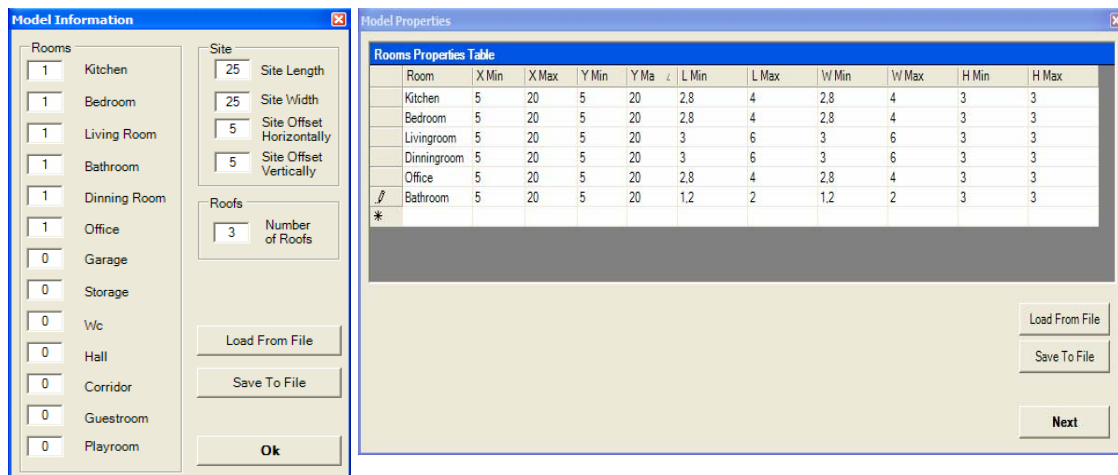


Figure 6-1 Model information

Figure 6-2 Room properties

Room Properties

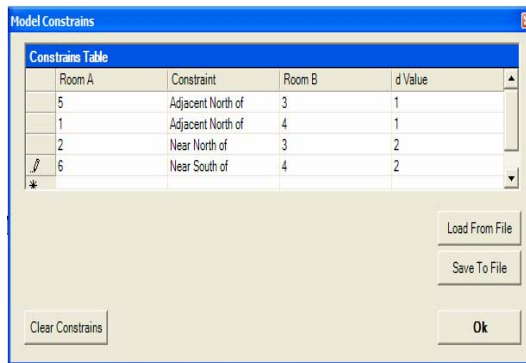
All the spaces have variable dimensions. The range of their dimensions is introduced by the designer under the requirements of the brief. In Figure 6-2 shows the ranges for the placement and magnitudes for the six rooms.

Room Constrains

The building brief expresses numerous demands that they entered in the system in the form of *binary relationships*. Such kind of relations controls and constrains the relative placement of spaces within the building. The main type of these relations expresses topological relations between the spaces. These are playing the role of *hard constraints* and they could not (and would not) get violated during the run of the genetic algorithm. Four of such relations are defined as shown in Figure 6-3.

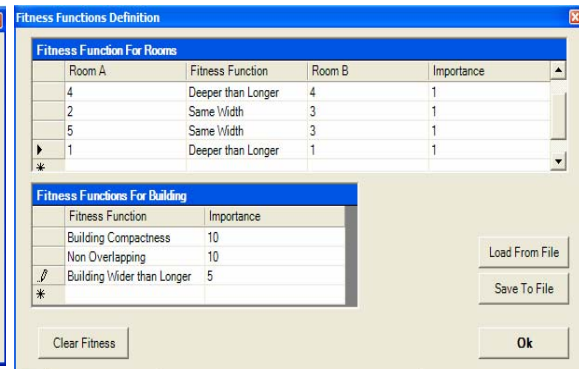
6.2.1.1.2. Room and Building Objective Functions

The specific design *principles* of a style could appeared in two separate sets as *local* and *global* evaluation criteria. That separation provides a number of objective functions for the spaces, and a number for the total building.



| Room A | Constraint | Room B | d Value |
|--------|-------------------|--------|---------|
| 5 | Adjacent North of | 3 | 1 |
| 1 | Adjacent North of | 4 | 1 |
| 2 | Near North of | 3 | 2 |
| 6 | Near South of | 4 | 2 |

Figure 6-3 Room constraints



| Room A | Fitness Function | Room B | Importance |
|--------|--------------------|--------|------------|
| 4 | Deeper than Longer | 4 | 1 |
| 2 | Same Width | 3 | 1 |
| 5 | Same Width | 3 | 1 |
| 1 | Deeper than Longer | 1 | 1 |

| Fitness Function | Importance |
|----------------------------|------------|
| Building Compactness | 10 |
| Non Overlapping | 10 |
| Building Wider than Longer | 5 |

Figure 6-4 Rooms fitness functions –
Building fitness functions

Room Fitness Functions

These criteria have a local character, while they influence both the dimensions and the placement of the individual spaces. Santorini style is characterised by long façade buildings. Therefore, the style includes two objective functions in order to *press* for two rooms to became *Deeper than Long*. For this reason in the particular set of style objectives it is introduced twice the type of *Longer than Deep*. Another important objective for that style is that the rooms have the same width as possible. Therefore the objective *Same Width* it is used for two consecutive rooms. The *weight* of importance of all local objectives is set to 1, (Figure 6-4).

Building Fitness Functions

These criteria have a global character and they influence both dimensions and placement of the individual spaces. The *Santorini* style has a distinct character that it is wider than long. This characteristic is provided in the synthesis of the spaces of typical buildings that belong in the specific style. For that reason the characteristic type of building fitness function is *Building Wider than Long* with importance weight set to 5. The other two objective functions are *Building Compactness* and *Non*

Overlapping, which in turn they both characterise that regional style as it is described in chapter 3; the importance weight is set to 10 (Figure 6-4).

6.2.1.1.3. Experiment – A

For the implementation of the genetic algorithm the designer could follow some pre-specified properties. However it is possible that the designer could redefines such parameters that control the implementation of the genetic algorithm in order to adapt better to a particular design problem. We underline the impact of such setting because it enables an improved performance for the genetic algorithm. Although the designer could not alter the settings of a style, he/she could take advantage of alternative search for the solution space by the genetic algorithm.

Genetic Algorithm Properties

The algorithm was run for 5 consecutive times, each time with different initial generation and for 1200 generations. The number of population is 200. The rate of mutation is set to 9%. A number of 20 elitist individuals that they ‘survive’ for 5 consecutive generations before they stop participate in the evolution (Figure 6-5).

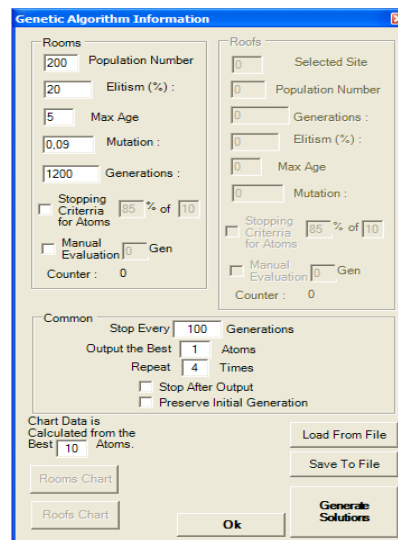
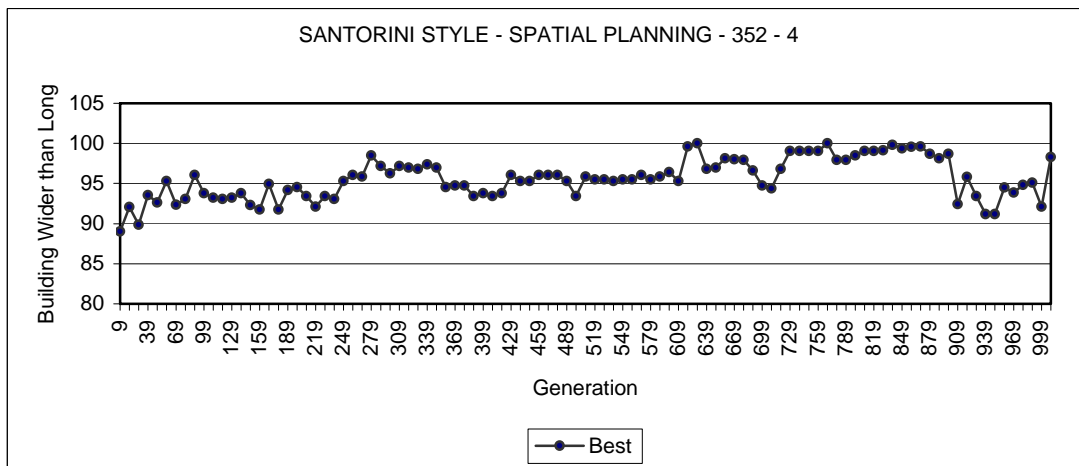
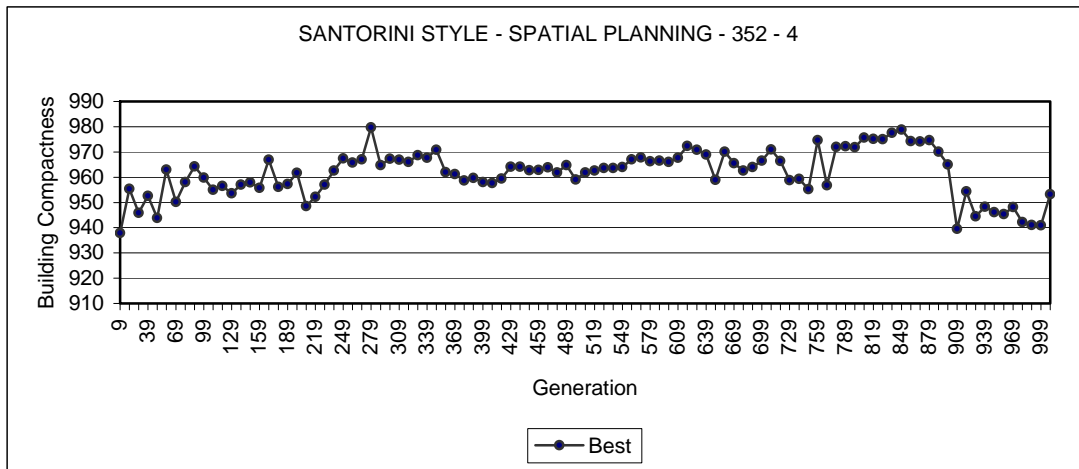


Figure 6-5 Genetic Algorithm Information

Result

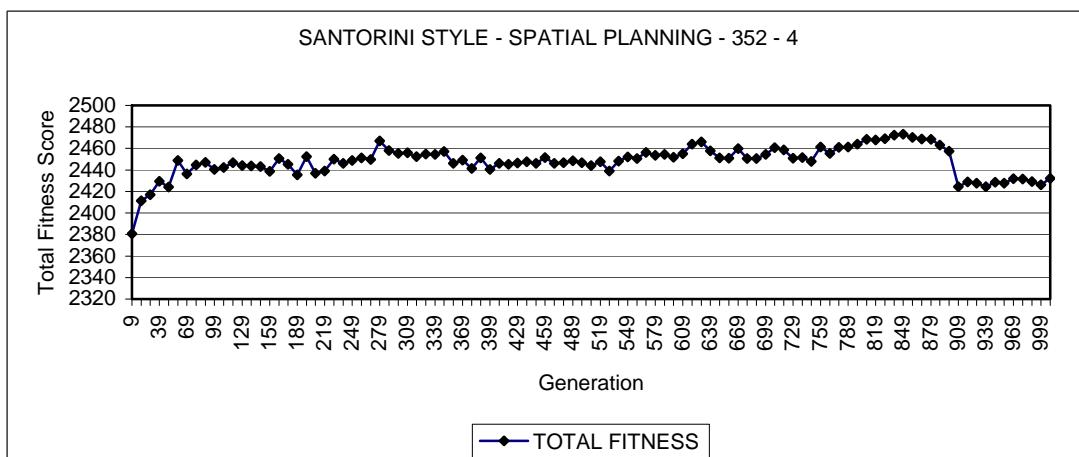
We provide the development of two of the seven objective functions of this style, *Building Compactness* and *Building Wider than Long*. The following two charts present the two normalised objective functions (as explained in Chapter 4). While the

genetic algorithm is not very stable we obtain improved individuals representing acceptable building designs for the soft constraints of the style.



Total fitness score charts

The following chart presents the total seven normalised objective functions (Chapter 4).



As it is shown the evolution of the total score of the best individual it is moving towards better figures. The examination of the progress of evolution for the building designs provides that during the first generations the stylistic objectives are not yet satisfied while the designs are in accordance with hard constraints. After a number of generations on the evolution provides designs more closely to the stylistic criteria.

The following table presents a set of design examples from different generations along the run of the particular case study. We can have an idea that while the evolution is directed at the same time the MOGA explores a great space of solutions and it is not trapped within local optimal solutions.

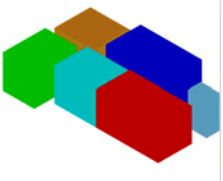

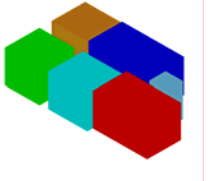


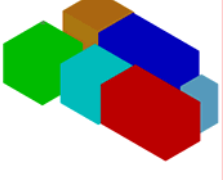
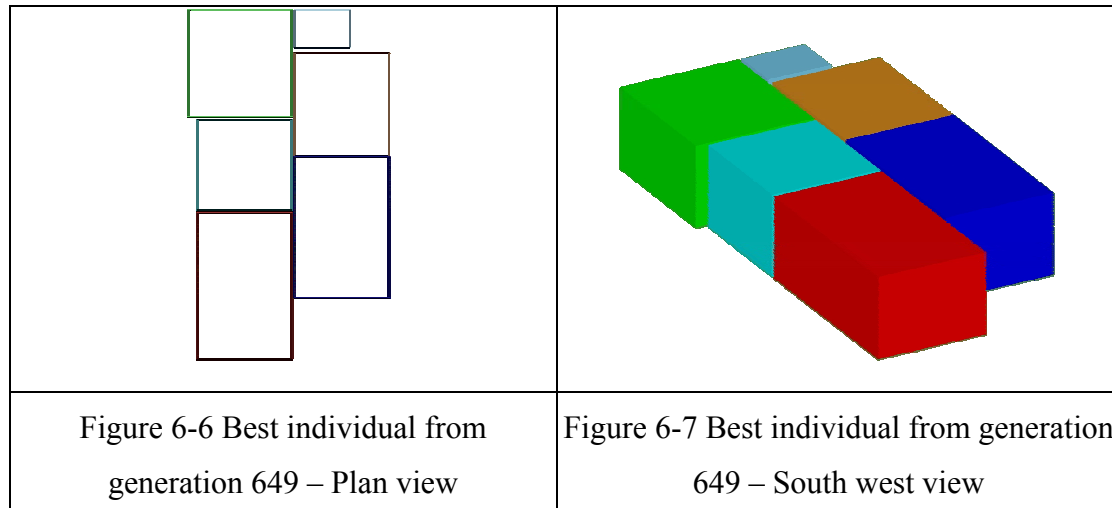
| | | |
|---|---|---|
| Repeat time 4 Generation 49 | Repeat time 4 Generation 249 | Repeat time 4 Generation 449 |
|  |  |  |
| Repeat time 4 Generation 649 | Repeat time 4 Generation 849 | Repeat time 4 Generation 1049 |
|  |  |  |

Table 6-1 Spatial planning examples from different generations

Scene solution A

In the following figures it is presented the best individual of generation number 649, during loop 4, from the *Scene 352*. The specific building synthesis is very well adapted to the objectives of the Santorini style (Figure 6-6). In particular, the spatial planning of the spaces respects all hard constraints as the later introduced by the designer. As it concerns the principles of Santorini style that evaluate the development of these designs, we can observe that the requested rooms have same width, other rooms are deeper than longer. Furthermore, the whole building synthesis has

dimensions that are deeper than long. The final design synthesis is very well adapted to the Santorini style, while it provides a slightly different composition which it is still characteristic of that style (Figure 6-7).



6.2.1.2. Scene B

These results concerns the run of *Scene 350* that has exactly the same building setup of the earlier *Scene A* description, (*Scene 352*). In particular, firstly appears Scene solution *B-1* of the best individual from generation 969 as evolved during loop 2. Secondly, is presented Scene solution *B-2* of the best individual from generation 729 as evolved during loop 3.

6.2.1.2.1. Experiment B

We present two examples of Scene solutions of successful designs from two alternative runs of five repetition loops. The genetic algorithm for every loop did not use the same initial generation.

Genetic Algorithm B-1

The algorithm was run for 5 consecutive times, each time with different initial generation. The number of individuals in the population was 200. *Scene 350* have the same settings except for the rate of mutation (8%) and number of generations (1000). A number of 20 elitist individuals they *survive* for five consecutive generations before they stop participate in the evolution (Figure 6-8).

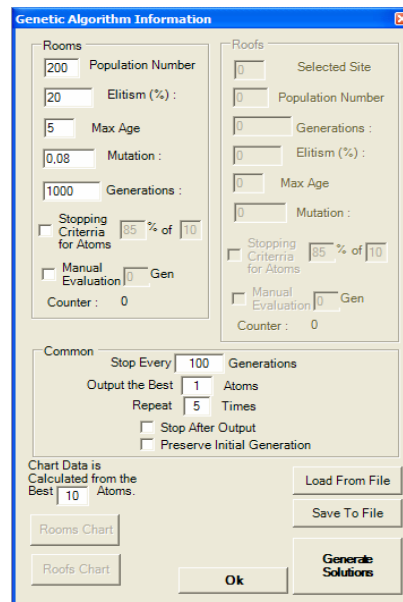
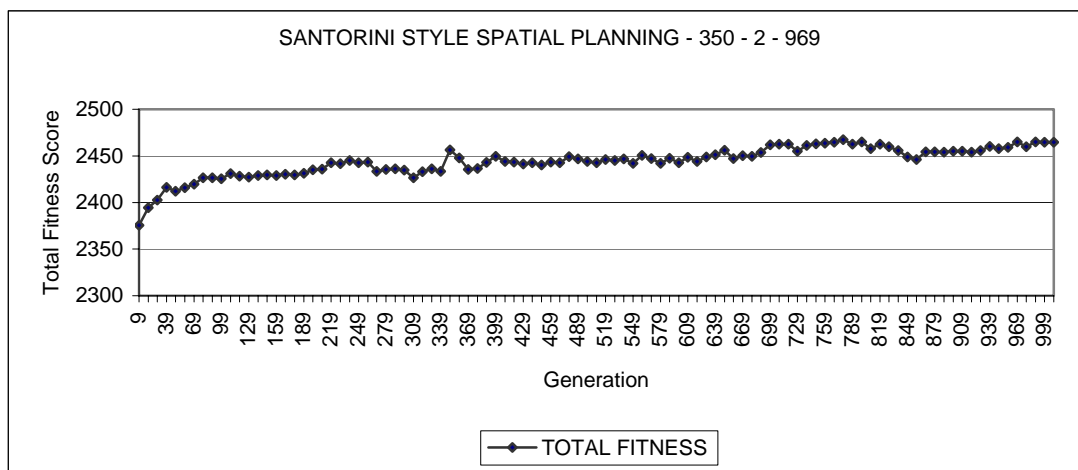


Figure 6-8 Genetic Algorithm Information

Result

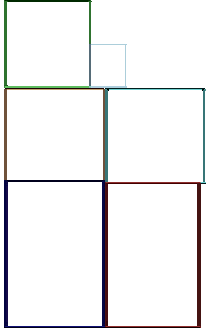
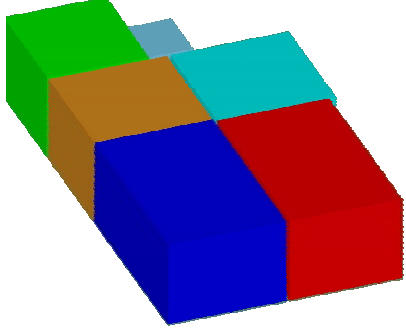
We present the total seven objective functions of Santorini style. The chart shows information for every ten generations along the 1000 total generations. The following chart presents the total seven normalised objective functions (Chapter 4). The evolution of the best individual it is moving again towards better figures.



Scene Solution B-1

In the following figures it is presented a best individual of the current run of 1000 generations. The chosen individual is the best individual of generation number 969. In Figure 6-9 it is presented the building in plan view. It is clear that the specific building

synthesis is very well adapted to the objectives of the Santorini style. In particular, the spatial planning of the spaces respects all hard constraints as the later introduced by the designer. All the requested topologic relationships between rooms are remain valid and were satisfied. As it concerns the principles of Santorini style that evaluate the development of these designs, we can observe that the requested rooms have same width, while the requested rooms are deeper than long – in this case study the living room (bleu) and the dinning room (red). Furthermore, the whole building synthesis has dimensions that are deeper than long, a definite characteristic of the Santorini architectural style. The specific example shows that the final design synthesis is very well adapted to the Santorini style. However, the final design provides a slightly different composition which it is still characteristic of that style (Figure 6-10).

| | |
|--|---|
|  |  |
| <p>Figure 6-9 Best individual from generation 969 – Plan view</p> | <p>Figure 6-10 Best individual from generation 969 – South west view</p> |

Genetic Algorithm B-2

The algorithm was run for 5 consecutive times, each time with different initial generation. The number of individuals in the population was 200. The rate of mutation was set up to 8%. There were a number of 20% elitist individuals that they ‘survive’ for 5 generations before they stop participate in the evolution. All these setups came from the developer and the designer has no further interaction with the MOGA part of the system. The genetic algorithm was run for 1000 generations (Figure 6-11).

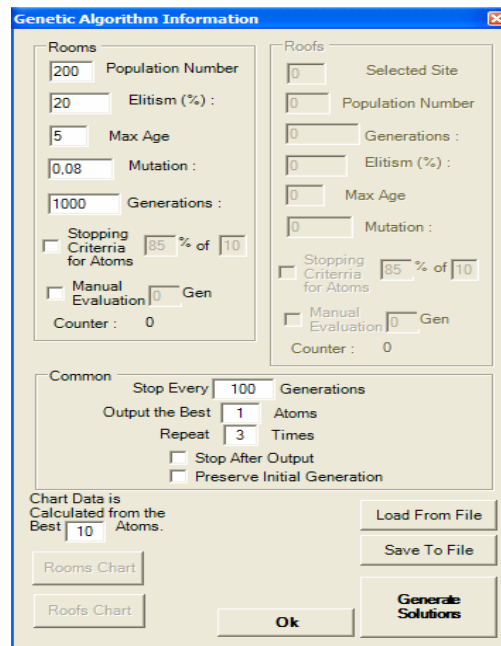
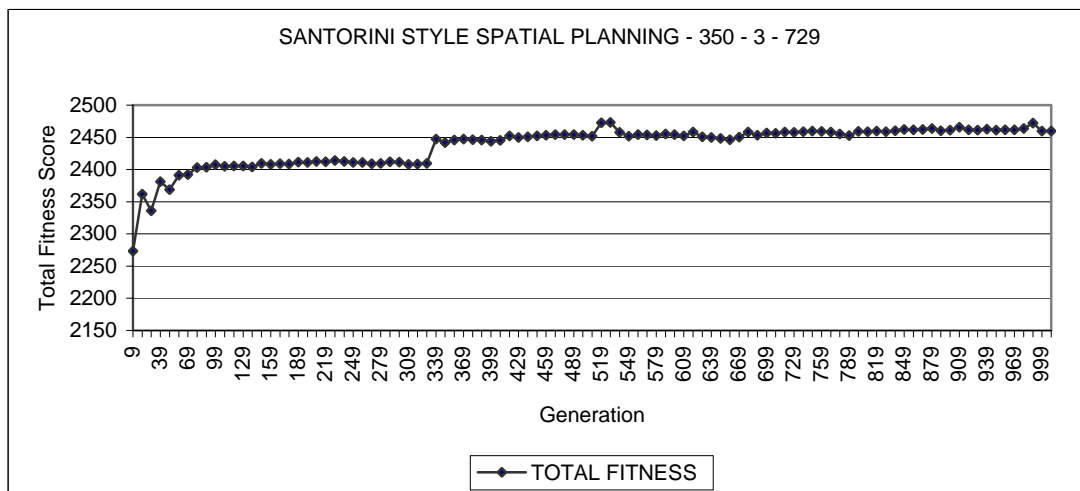


Figure 6-11 Genetic Algorithm Information

Results

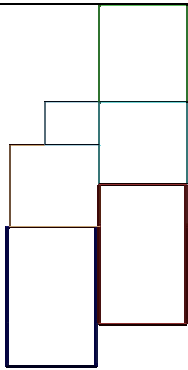
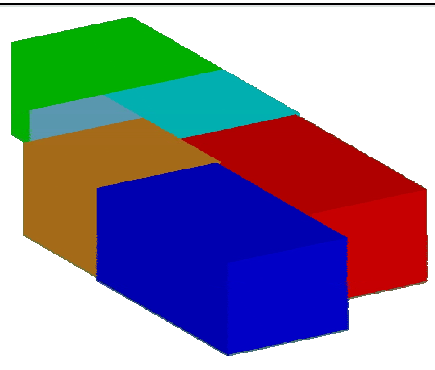
We present the development of two of the seven objective functions of Santorini style. In the following chart it is presented the total seven normalised objective functions (chapter 4).



As it is shown from the chart again the evolution of the total score of the best individual it is moving towards better figures after the 430th generation. Generation after generation we obtain improved individuals representing acceptable building designs for the given hard constraints and soft constraints.

Scene solution B-2

In the following figures it is presented the best individual of generation number 729 from loop 3 of the *Scene 350*. It is clear that the specific building synthesis is very well adapted to the objectives of the Santorini style. In particular, the spatial planning of the spaces respects all hard constraints as the later introduced by the designer, (Figure 6-12). The requested topologic relationships between rooms are remain valid and were satisfied. Furthermore, the building syntheses are well adapted to the principles of Santorini style, (Figure 6-13). However, the final design provides a slightly different composition which it is still characteristic of the style.

| | |
|--|---|
|  |  |
| <p>Figure 6-12 Best individual from generation 729 – Plan view</p> | <p>Figure 6-13 Best individual from generation 729 – South west view</p> |

6.2.2. Roof Morphology

In the second step of the experiment, we evaluate the performance of the Evolutionary-MultiCAD system during the attempt to search for the morphology of roof(s). In the current experiment we provide results concerning the evolution of building designs of given requirements, under specific morphological principles of the Santorini architectural style. A number objective functions as expression of stylistic principles direct the evolution of the individuals in order for the later to have roofs adapted to the Santorini architectural style. After the completion of a run considering the spatial planning of the building, the designer proceeds towards the process of entering appropriate information for the development of the roof(s) morphology. Given a number of roof requirements from the part of the designer the input process has the following steps along the Evolutionary-MultiCAD system:

- Selection of a building model from the set of models from the last generation of the MOGA. The system proceeds by placing a building model in the site, while all relevant information of the spaces is available to be associated with any of the roof(s).
- Definition of which space is covered by which roof according to number of roofs that were declared in the beginning of the process. As a consequence a roof inherits the dimensions of the space or spaces, which it is associated with.
- Selection of the module of an appropriate style that the designer wants to apply for the evolution of the system.
- Definition of the parameters that control the implementation of the genetic algorithm. The designer could follow some pre-specified properties for the implementation of the genetic algorithm. However it is possible that the designer could redefines such parameters that control the implementation of the genetic algorithm in order to adapt better to a particular design problem.

6.2.2.1. Scene A

The objects of *Scene A*, properties and relations are considered as hard constraints. The stylistic criteria are considered as soft constraints and are the evaluation criteria. We select from the previous step to search the roof morphology for a building from *Scene 350*. We select the best Scene from generation 969 during loop 2.

6.2.2.1.1. Definition of Roof(s)

The designer has already defines the number of roofs for the building. In this design case the number of roofs is three (Table 6-2).

| | |
|-----------------|---|
| Number of Roofs | 3 |
|-----------------|---|

Table 6-2 Number of Roofs

Room Properties

The designer selects a building composition produced from the earlier run of the MOGA. Consequently, all the spaces have specific placement positions and specific dimensions, (Table 6-3).

| Type | Placement | | Dimensions | | |
|--------------|-----------|------|------------|-----|-----|
| | X | Y | L | W | H |
| Kitchen | 16 | 15,2 | 3,5 | 3,7 | 3 |
| Bedroom | 12,4 | 19 | 3 | 3,4 | 3 |
| Living room | 12,4 | 9,5 | 3,5 | 5,8 | 3 |
| Dinning room | 16 | 9,5 | 3,3 | 5,7 | 3 |
| Office | 12,4 | 15,3 | 3,5 | 3,6 | 3 |
| Bathroom | 15,4 | 19 | 1,3 | 1,7 | 3 |

Table 6-3 Room placement - dimensions

Roof Properties

All roofs have specific dimensions inherited by the spaces that they will cover. In particular, the length and width of a roof are both fixed and are the resulted length and width from the bounding box of the spaces that the roof covers. The roof height is variable.

Roof Constrains

The designer defines as hard constraints which roof covers which type of spaces of a given building composition. A roof could cover as a minimum one space. These constraints are applied in the form of binary relationships between the roof and a space and as hard constraints they would not get violated during the run of the GA. For the current design case the relations between the three roofs and the six spaces is presented in the following table (Figure 6-14).

| Roof | Covers | Room |
|------|--------|------|
| 2 | Covers | 1 |
| 1 | Covers | 2 |
| 1 | Covers | 3 |
| 2 | Covers | 4 |
| 2 | Covers | 5 |

IMPORTANT!! Rooms that share the same Roof must be Adjacent.

Figure 6-14 Roofs relations

6.2.2.1.2. Roof Objective Functions

The Santorini style has a distinct character concerning roof(s) morphology. The morphology concerns types of semi barrel-vaulted roof and simply flat roof. Therefore the appropriate objective functions are applied for each roof separately. For the two vaulted roofs, the style provides an objective functions in order a roof(s) to become parallel with the greatest dimension of the building. In this case the greatest dimension is its depth. The fitness is *Parallel to Longer dimension*. The other objective function is *Roof Consistency*. The latter directs the evolution of the roof towards consistent and symmetric morphology. The fitness function *Maximise W1 W3* has importance weight set to 4. The other two objective functions are *Parallel to Longer Dim* and *Roof Consistency*. Both they directs the evolution of the roof towards consistent and symmetric morphology.

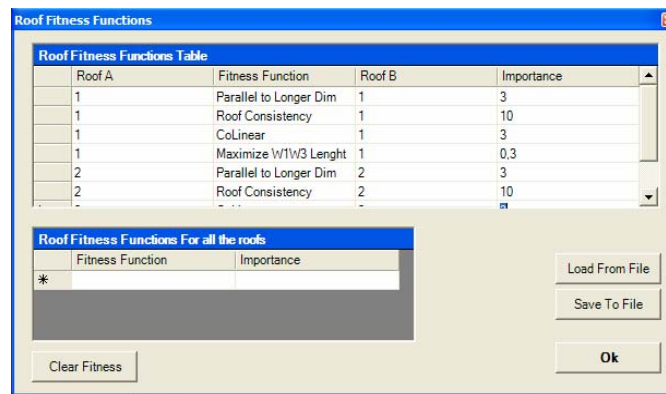


Figure 6-15 Roof Fitness Functions

The following two criteria, *Minimise WIW3 Length* and *Collinear*, evaluate the geometry of the roof's form towards the type of *vaulted* morphology (Figure 6-15). In order to have a flat form except the *Roof Consistency* two more objective functions are needed, *Minimise WIW2 Length* and *Minimise W3W2 Length* they both evaluate the degree of flatness for the third roof, (Figure 6-15)

6.2.2.1.3. Experiment A

For the implementation of the GA the designer follow some pre-specified properties. However it is possible that these parameters could be redefined for better adaptation to a particular design problem.

Genetic Algorithm Properties

The GA was run for 5 times, each time with different initialisation for 800 generations. The individuals in the population were 100. The mutation was 2%. Elitist individuals are 20% and they *survive* for 5 generations (Figure 6-16).

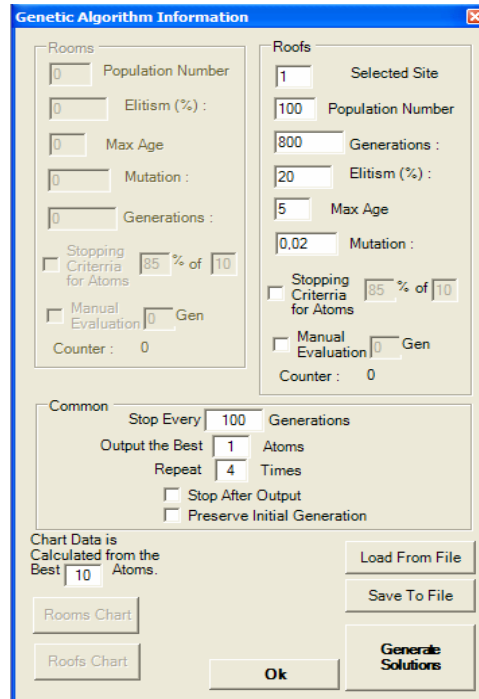
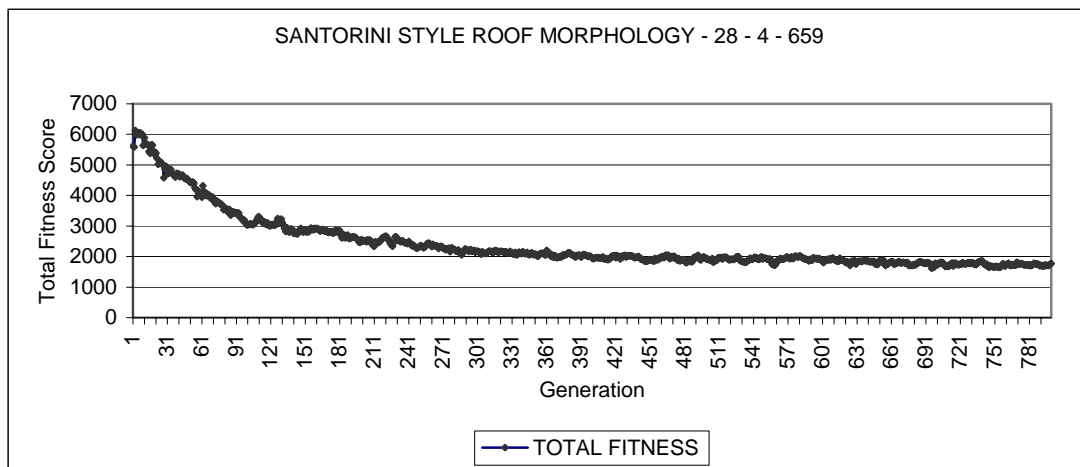


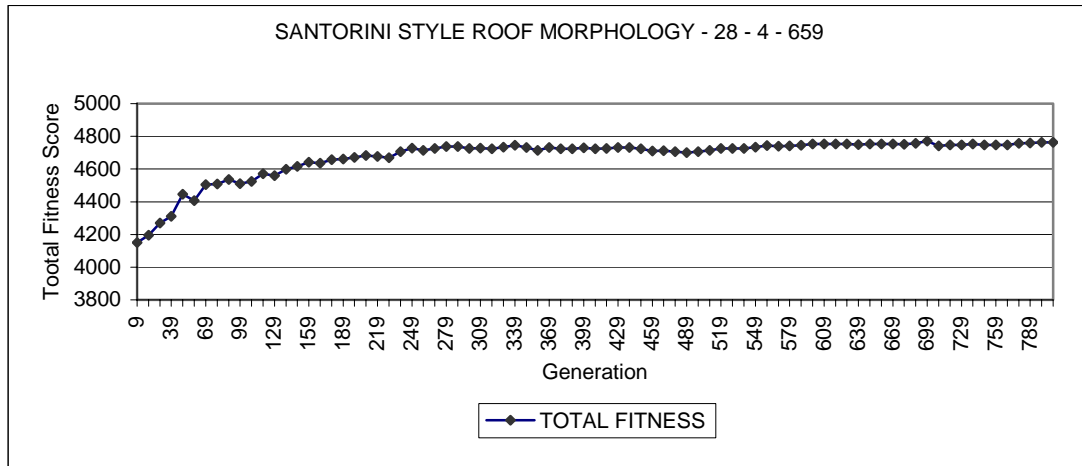
Figure 6-16 Genetic Algorithm Information

Results

The following charts provide the eleven objective functions of this style for the roof development. In particular, it is presented the sum of all objective functions, for the best individual from all generations. The resulted scores for the best individual are progressive high and remain high till the end of evolution.



Therefore, we obtain improved individuals representing acceptable roof forms for the given hard constraints of the description and the soft constraints of the specific style. In the following chart it is presented the same total score as a sum of objective functions normalised (chapter 4).



The following table (Table 6-4) presents a set of Roof examples from different generations along the run of the particular case study.

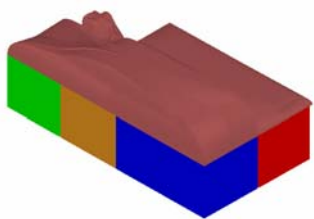
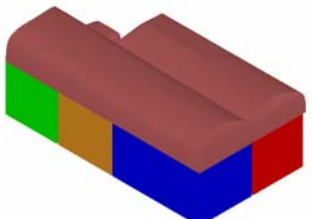
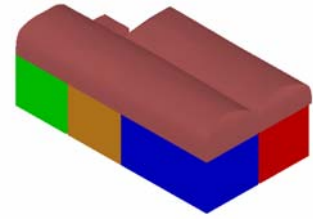
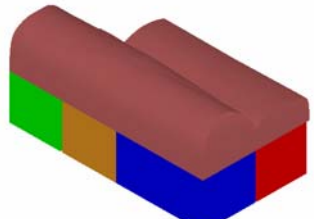
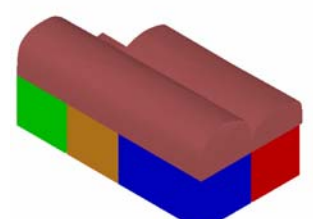
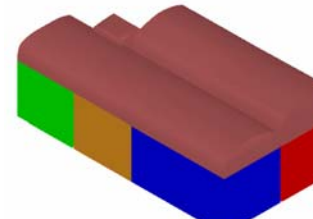
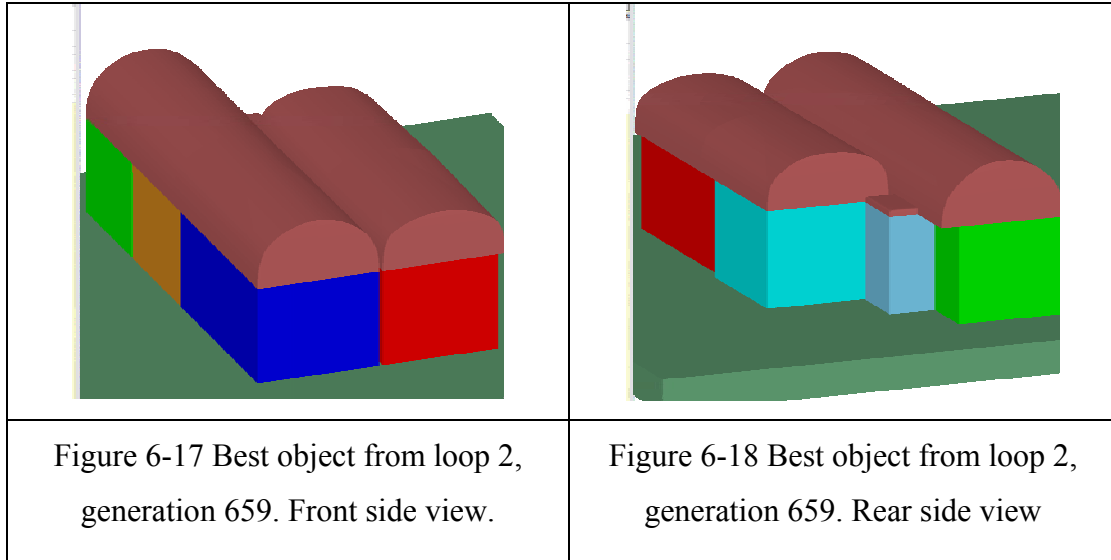
| | | |
|---|---|--|
| Repeat time 2 Generation 9 | Repeat time 2 Generation 259 | Repeat time 2 Generation 459 |
|  |  |  |
| Repeat time 2 Generation 659 | Repeat time 2 Generation 759 | Repeat time 2 Generation 800 |
|  |  |  |

Table 6-4 Roof examples from different generations

Scene solution A

We present the best individual of generation number 659 (loop 4) from *Scene 28*. The roofs are very well adapted to the objectives of the Santorini style, (Figure 6-17).



Roof-1 and *Roof-2* has taken the characteristic *vaulted* shape of Santorini style. It is important to underline that both shapes have an approximate *vaulted* form, an observation that it is also appeared in the real buildings under that style. We consider of great interest the fact that the morphology of the two roofs is not exactly the same. This fact ensures diversity between the resulted designs from the genetic algorithm. In the next figures (Figure 6-18 and Figure 6-19) we present a perspective view from the south west side of the building.

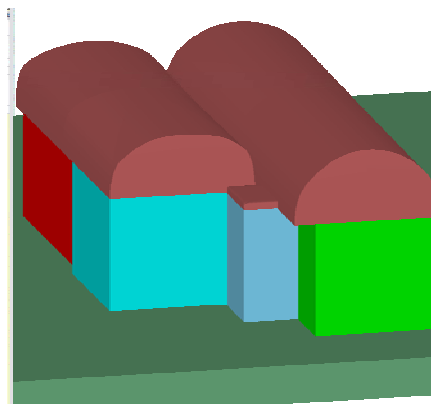


Figure 6-19 Best object from loop 2, generation 659. Rear side view

6.2.2.2. Scene B

The selected Scene from the spatial planning stage is the best individual from generation 729 as evolved during loop 3 from *Scene 350*.

6.2.2.2.1. Definition of Roof(s)

The designer has already defined the number of roofs that he/she wants for the building. In this design case the number of roofs is three (Table 6-5).

| | |
|-----------------|---|
| Number of Roofs | 3 |
|-----------------|---|

Table 6-5 Number of Roofs

Room Properties

The designer selects a building composition produced from the earlier run of the MOGA (Table 6-6).

| Type | Placement | | Dimensions | | |
|--------------|-----------|----------|------------|----------|----------|
| | <i>X</i> | <i>Y</i> | <i>L</i> | <i>W</i> | <i>H</i> |
| Kitchen | 17,2 | 13,3 | 3,1 | 3,3 | 3 |
| Bedroom | 17,2 | 16,6 | 3 | 3,9 | 3 |
| Living room | 14 | 6 | 3,1 | 5,6 | 3 |
| Dinning room | 17,2 | 7,7 | 3 | 5,6 | 3 |
| Office | 14,1 | 11,6 | 3,1 | 3,3 | 3 |
| Bathroom | 15,3 | 14,9 | 1,9 | 1,7 | 3 |

Table 6-6 Rooms placement - dimensions

Roof Properties

All the roofs dimensions are inherited by the spaces that they cover, as explained earlier. The height of a roof is variable.

Roof Constrains

The designer defines as hard constraints which roof covers which number of spaces of a given building composition. The relations between the three roofs and the six spaces are presented in the following table (Figure 6-20).

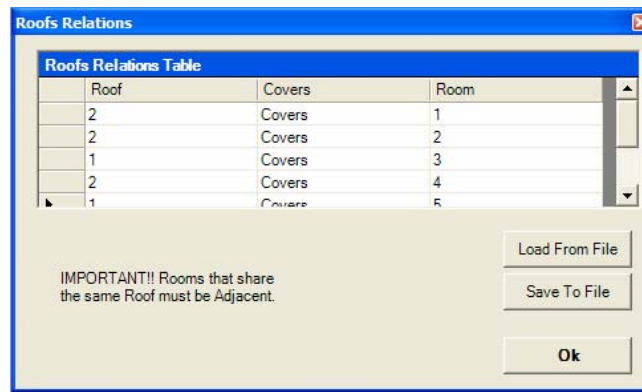


Figure 6-20 Room – Roof Relations

6.2.2.2.2. Roof Objective Functions

We apply the appropriate objective functions as earlier for each roof separately, (Figure 6-21). The general fitness for all roofs is *Parallel to Longer dimension* and *Roof Consistency*. The fitness function *Maximise $W1 W3$* , *Minimise $W1W3$ Length* and *Collinear* evaluate the geometry of the roof's form towards vaulted morphology. For the third flat forms roof both *Minimise $W1W2$ Length* and *Minimise $W3W2$ Length* are objective criteria that evaluate the degree of flatness.

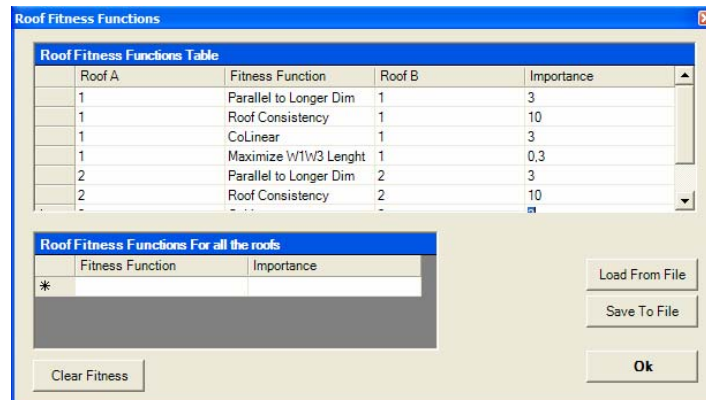


Figure 6-21 Roofs Fitness Function

6.2.2.2.3. Experiment B

For the experiment B we follow the same guidelines for the setting of the GA.

Genetic Algorithm Properties

The GA was run for 5 times, each time with different initialisation for 800 generations. The population was 100. The rate of mutation was 2%. The number of elitist individuals was 20 (Figure 6-22).

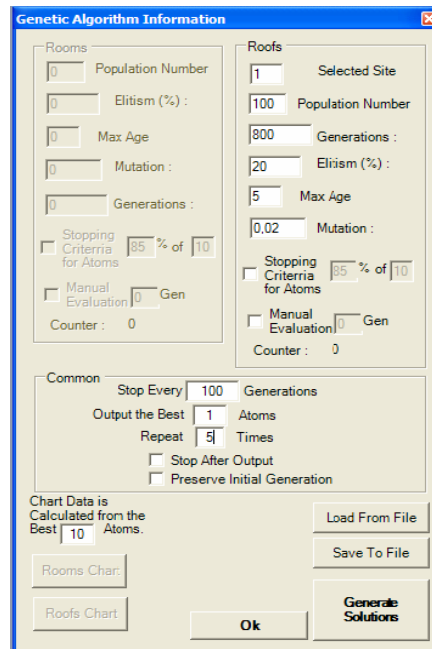
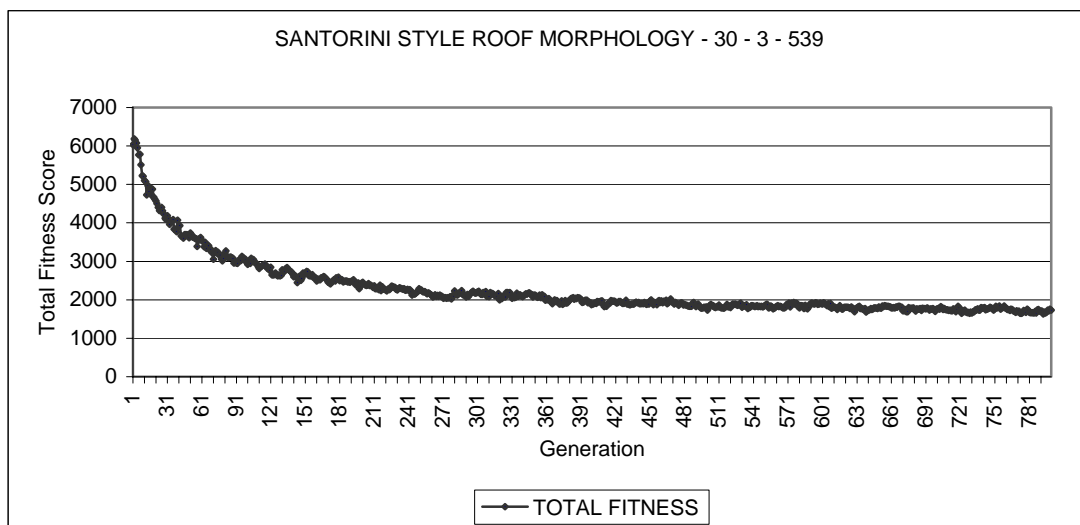


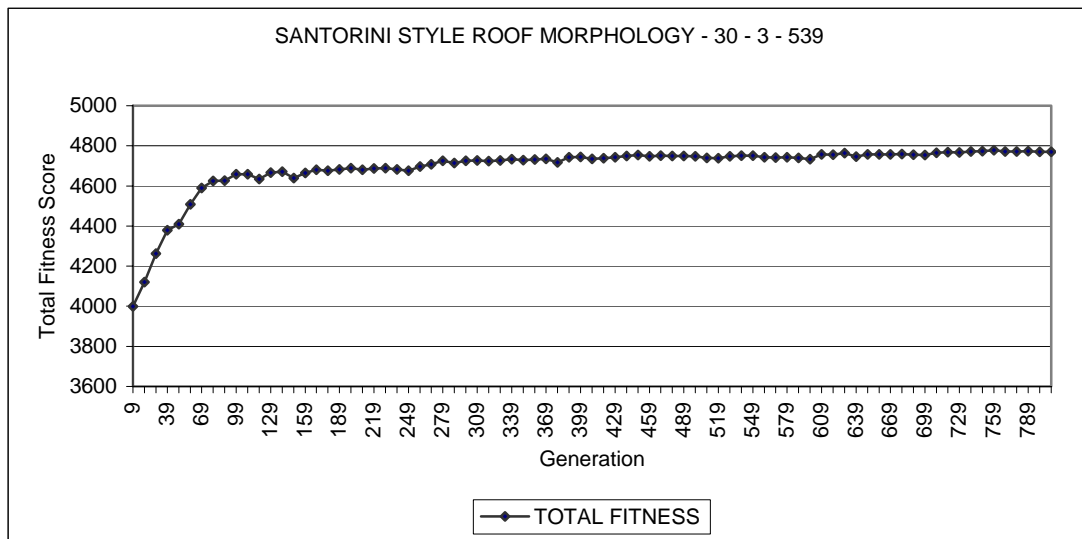
Figure 6-22 Genetic Algorithm Information

Results

The following charts provide information for the evolution of roof morphology. We provide the development of the eleven objective functions of this style. In particular, it is presented the sum of all objective functions, for the best individual from all generations. The resulted scores for the best individual are progressive high and remain high till the end of evolution. Therefore, we obtain improved individuals representing acceptable roof forms.



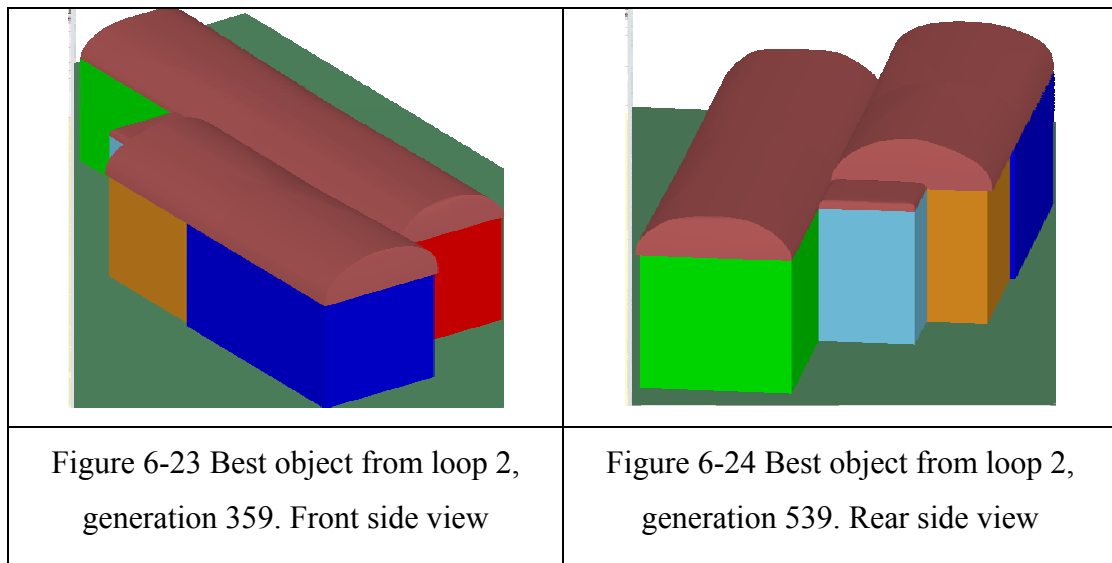
In the following chart it is presented the same total score as a sum of normalised objective functions.



As it is shown from the chart, after the 100th generation the GA is more stable and we obtain improved individuals representing acceptable roof forms for the given soft constraints of the specific style.

Scene solution B

The best individual is presented from generation 539, (loop 3) of *Scene 19*, (Figure 6-23). The roof morphology is very well adapted to the objectives of the Santorini style. *Roof-1* and *Roof-2* has taken the characteristic vaulted shape of Santorini style, (Figure 6-24).



It is important to underline that both shapes have an approximate vaulted form, a characteristic from real buildings under that style.

6.3. Metsovo Style

In this experiment we evaluate the performance of the Evolutionary Declarative design system when it is confronting simple buildings with no more than six spaces. We enable the spaces to have varied dimensions while they are only constrained by the building brief.

Stylistic principles expressed as objective functions direct the evolution of the individuals (scenes) in order for the later to express building designs adapted to the Metsovo architectural style. The whole process has two distinct steps. Firstly, the designer utilise the MOGA system in order to evolve the spatial planning for given building requirements, under specific principles of an architectural style. Secondly, from the results of the earlier step, in particular from the resulted individuals (scenes) of the last generation, the designer selects one individual in order to evolve the roof morphology of the building.

6.3.1. Space planning

In the first part of the current experiment we provide results concerning the evolution of building designs of given requirements, under specific compositional principles of the Metsovo architectural style. Given a number of building requirements from the part of the designer the input process has the same steps as in earlier experiments.

6.3.1.1. Scene A

We define now the appropriate objects of *Scene A*. All object properties and some relations are considered as hard constraints. the stylistic criteria are considered as soft constraints which are the evaluation criteria.

6.3.1.1.1. Definition of Scene

In this case a building configuration has six spaces and two roofs. The *Site* dimensions are 25 by 25 units, with an offset from each side by 5 units, (units are in meters). The quantification of site is set in steps of 0.10 meters. The *Number of Roofs* is 3, but this information will be used during the step of roof morphology, (Figure 6-25).

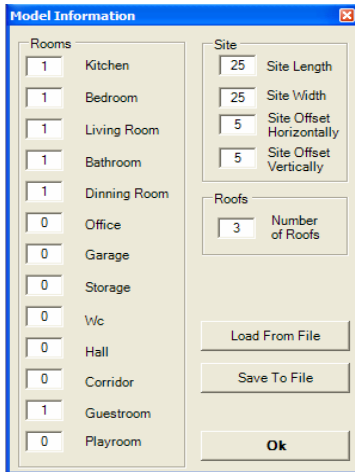


Figure 6-25 Model Information

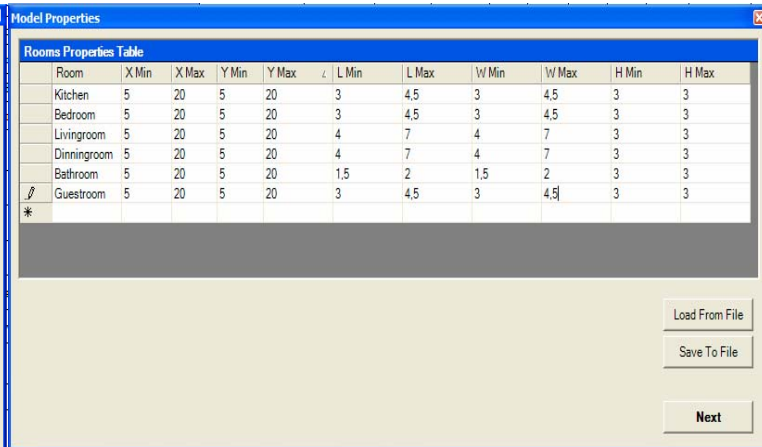


Figure 6-26 Room Properties

Room Properties

The spaces have variable dimensions as it shown in the following table (Figure 6-26). The range of their dimensions is introduced by the designer.

Room Constrains

The building brief expresses numerous demands that they entered in the system in the form of binary relationships. Such relations constrains the relative placement of spaces within the building (Figure 6-27).

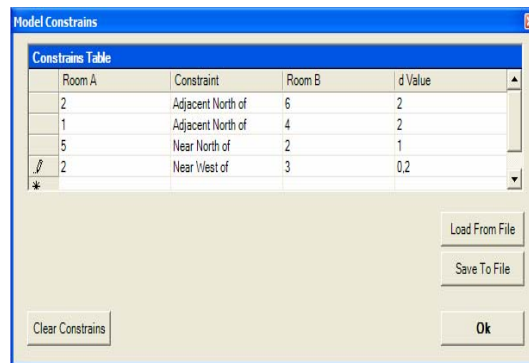


Figure 6-27 Rooms Constraints

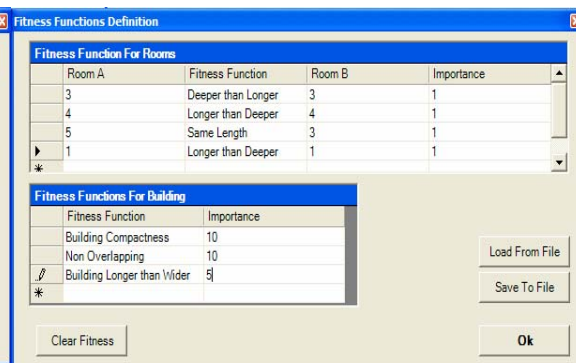


Figure 6-28 Building – Room Fitness functions

However such relations are hard constraints and they could not get violated during the run of the GA.

6.3.1.1.2. Room and Building Objective Functions

The specific stylistic principles appeared in two sets, local and global. That separation provides a number of objective functions for spaces, and a number for the building.

Room Fitness Functions

This style is characterised by long façade buildings and it includes an objective function in order for two rooms to become longer than wide. That set of objective (*Longer than Deep*) it is introduced twice (i.e. for two spaces), (Figure 6-28).

Building Fitness Functions

The main type of fitness function is *Building Longer than Wide* with importance weight set to 5. The other two objective functions are *Building Compactness* and *Non Overlapping* have importance weight set to 10 (Figure 6-28).

6.3.1.1.3. Experiment A

For the implementation of the genetic algorithm we follow again the general guidelines as presented during the experiment of Santorini style experiment.

Genetic Algorithm Properties

The algorithm is run for 5 times, (with different initialisation) for 1000 generations. The number of population is 100. The mutation is set up to 5%.

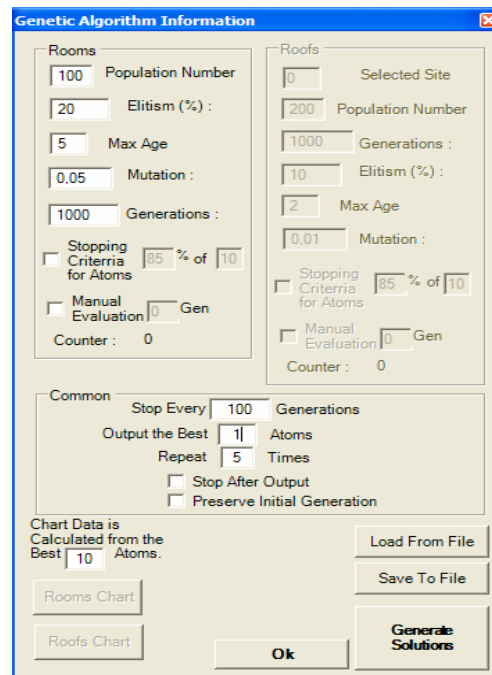
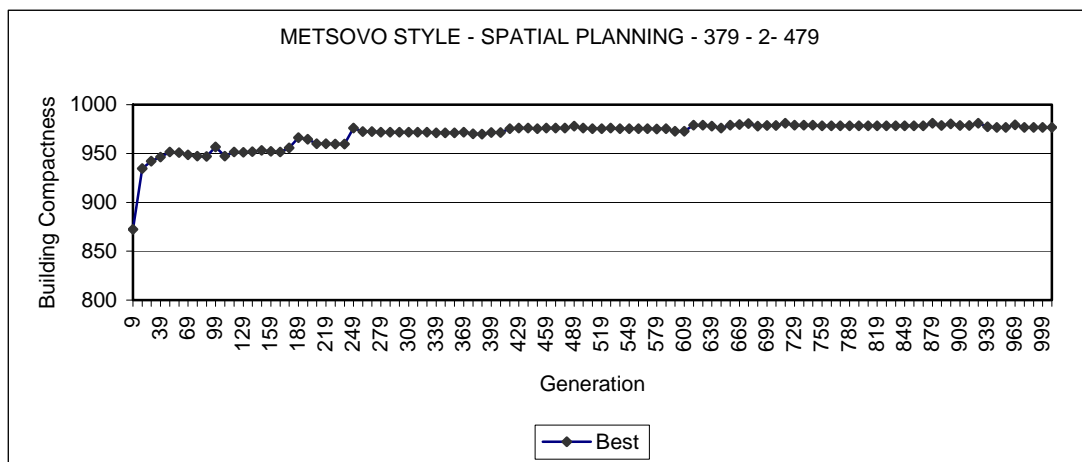


Figure 6-29 Genetic Algorithm Information

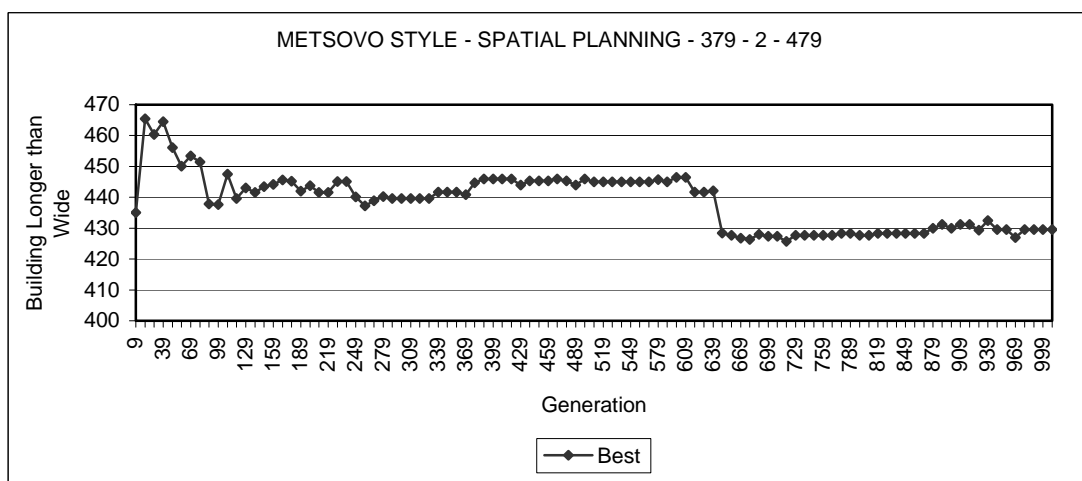
Elitist individuals are 20% of the population and they ‘*survive*’ for 5 generations, (Figure 6-29). The above GA parameters came from the developer. But the designer could redefine them in order to adapt better to a particular design problem.

Results

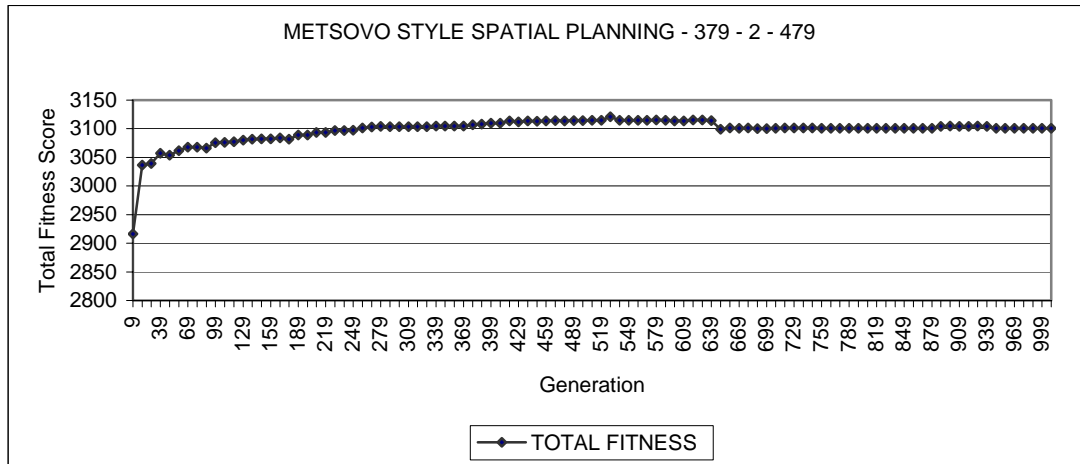
The following charts provide information for the evolution of building designs. We decided to provide for the current case study the development of two of the seven objective functions of this style, “*Building Compactness*” and “*Building Longer than Wide*”. The two objective functions are normalised (Chapter 4).



As it is shown from the chart we obtain improved individuals representing acceptable building designs for the soft constraints of the style after the 600th generation.



In the following chart it is presented the sum of total seven normalised objective functions (Chapter 4). The evolution of the total score of the best individual (as shown earlier too) it became more stable after the 100th generation. Approximately after the 640th generation the evolution provides designs more closely to the stylistic criteria.



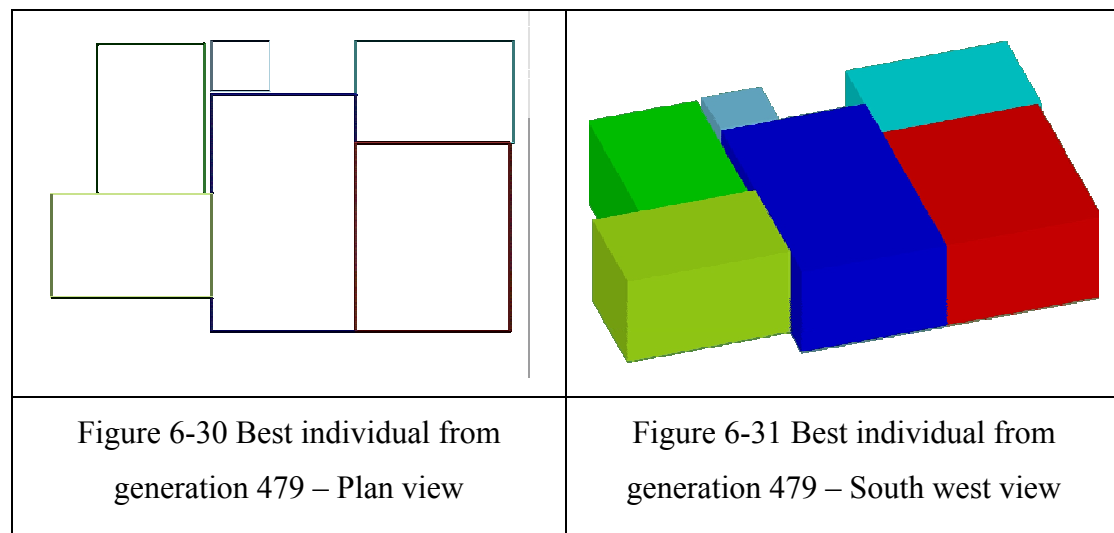
The following table (Table 6-7) presents a set of design examples from different generations along the run of the particular case study.

| | | |
|---------------------------------|---------------------------------|----------------------------------|
| Repeat time 2 Generation 79 | Repeat time 2 Generation 279 | Repeat time 2 Generation 479 |
| | | |
| Repeat time 2 Generation 679 | Repeat time 2 Generation 879 | Repeat time 2 Generation 1000 |
| | | |

Table 6-7 Spatial planning examples from different generations

Scene solution A

We present the best individual of generation 479 (loop 2) from *Scene 379*. The building synthesis is very well adapted to the objectives of Metsovo style (Figure 6-30). Furthermore, we can observe that some rooms have the same width, while other rooms are longer than deep. Among the important characteristics of that style is that at least two spaces of type *Public zone* are slightly out of the perimeter of a building. In this example such designs evolve with the bottom left spaces (blue and yellow), satisfying this stylistic principle. Furthermore, the building synthesis has dimensions that are longer than deep. The design is very well adapted to Metsovo style, while it provides a slightly different composition which it is characteristic of that style, (Figure 6-31).



6.3.1.2. Scene B

We define the objects of *Scene B* properties, relations. The results came from the run of another scene (*Scene 367*), which has exactly the same building setup of the earlier *Scene A* description (*Scene 379*). We show as Scene solution *B* the best individual from generation 829 (loop 1).

6.3.1.2.1. Definition of Scene

In this case the building composition has six spaces. The dimensions of the site are the same as earlier. The quantification of the *Site* is set in steps of 0.10 meters. The *Number of Roofs* is 2, (Figure 6-32).

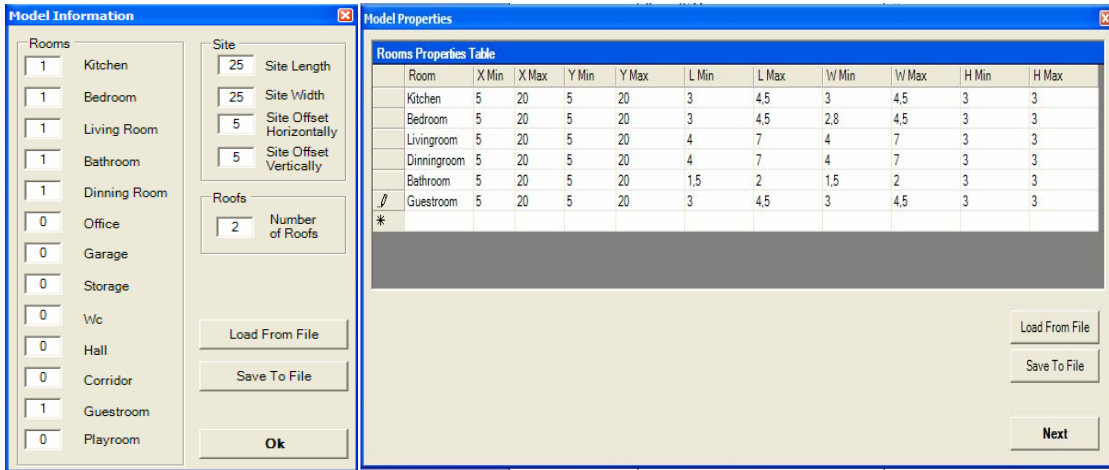


Figure 6-32 Model Information

Figure 6-33 Rooms Properties

Room Properties

All spaces have variable dimensions range (Figure 6-33).

Room Constrains

All building brief demands are considered as hard constraints and they would not get violated during the run of the GA (Figure 6-34).

6.3.1.2.2. Room and Building Objective Functions

The style principles appeared as, local and global, with a set of objective functions for spaces, and a set for total building respectively.

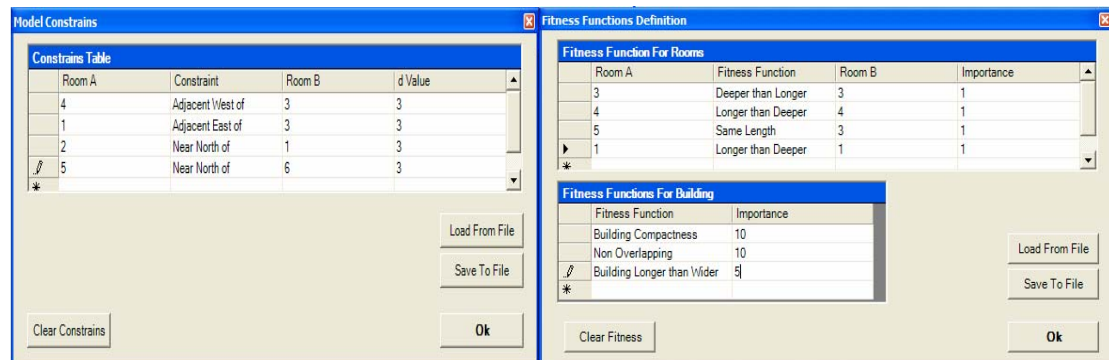


Figure 6-34 Rooms Constraints

Figure 6-35 Building – Rooms Fitness Function

Room Fitness Functions

This style is characterised by long façade buildings and has two objective functions in order to ‘*press*’ for two rooms to become longer than wide. For this reason it is introduced for two spaces, the type of *Longer than Deep* (Figure 6-35).

Building Fitness Functions

The characteristic type of building fitness function is *Building Longer than Wide* with importance weight set to 4. The objective functions *Building Compactness* and *Non Overlap*, have importance weight set to 10 (Figure 6-35).

6.3.1.2.3. Experiment B

We present two more examples of Scene solutions of successful designs from two alternative runs. The genetic algorithm’s preferences of *Scene 367* are the same with *Scene 379*.

Genetic Algorithm Properties

The population number is 150. The rate of mutation is 8%. Elitist individuals are 20% of the population and ‘*survive*’ for 5 generations before they exclude from evolution. The genetic algorithm runs for 1000 generations, (Figure 6-36).

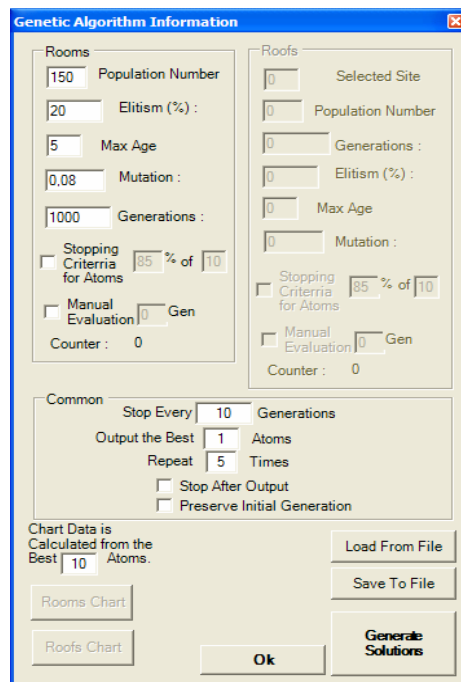
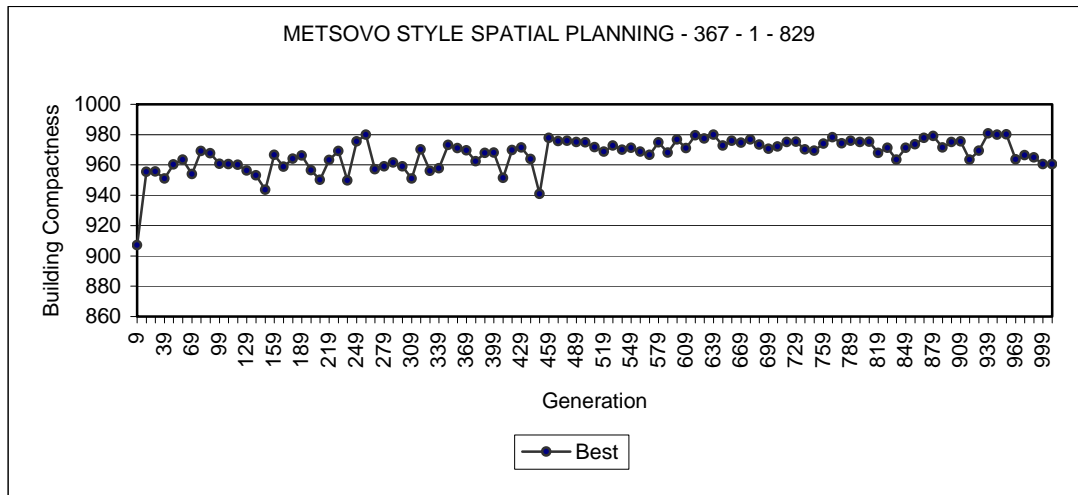


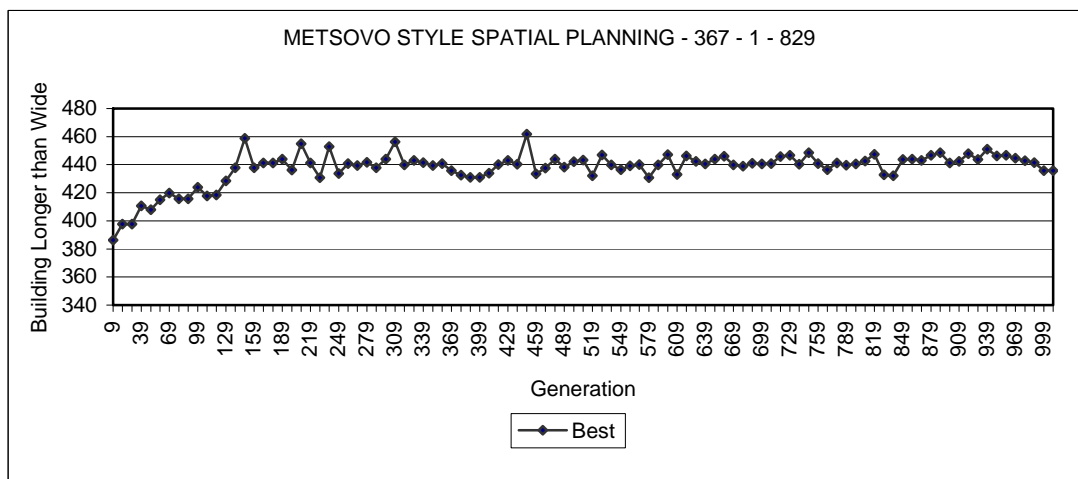
Figure 6-36 Genetic Algorithm Information

Results

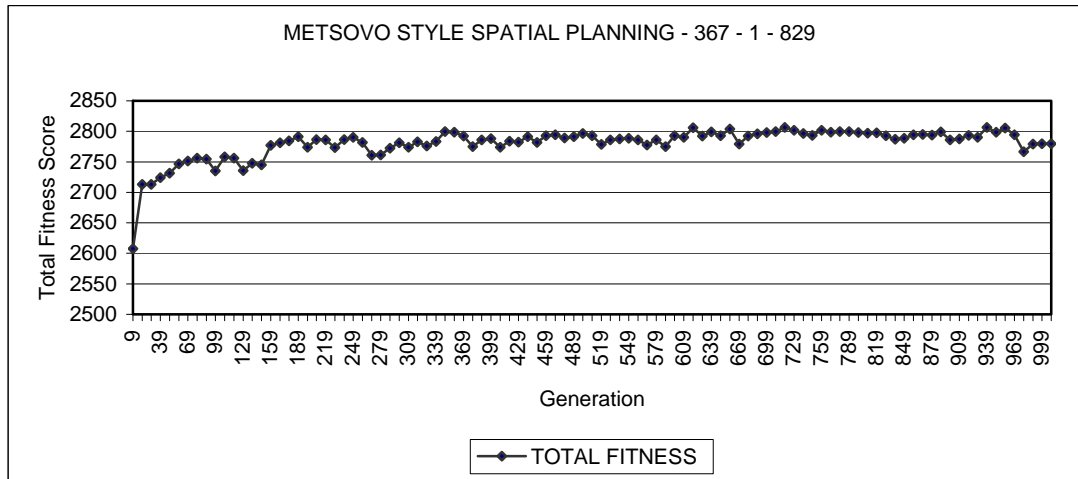
The following chart provides interesting information for the evolution of building designs. We provide the development of two of the ten objective functions of this style, *Building Compactness* and *Building Longer than Wide*.



In the next two charts it is presented the same two normalised objective functions (Chapter 4). As it is shown from the chart we obtain improved individuals after the 460th generation.

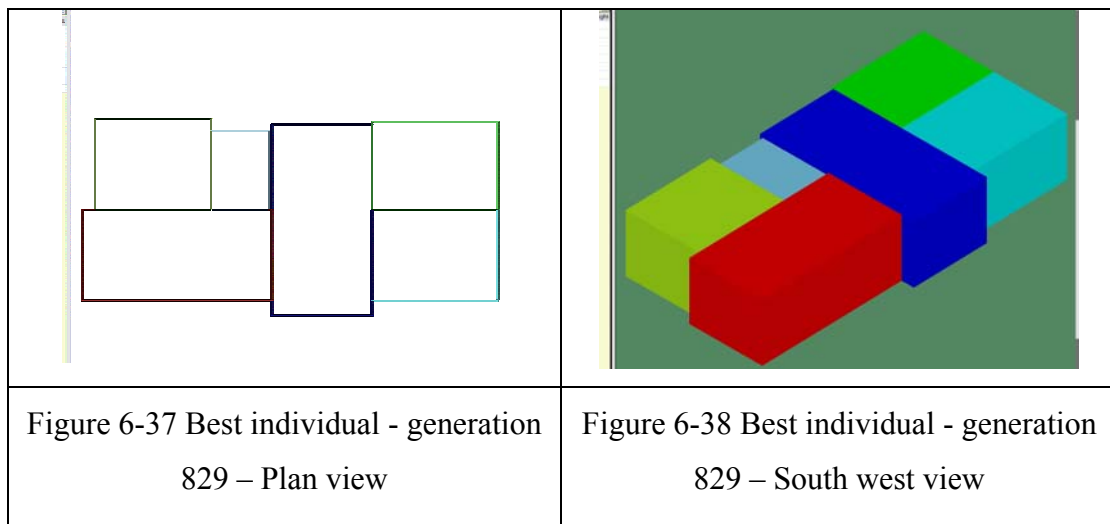


From both charts we observe that for both objectives until the middle of the evolution they are not very stable. However after the 610th generation approximately the algorithm converges. The following chart present the total normalised seven objective functions. the evolution of the best individual is moving towards better figures. Improved individuals are obtained representing acceptable building designs.



Scene solution A

The following figures present a best individual best individual of generation number 829, (loop 1) of *Scene 367*. The building is very well adapted to the objectives of the Metsovo style (Figure 6-37). The spatial planning respects all hard constraints. As it concerns Metsovo style, we observe that the requested rooms have *Same Width*, while other rooms are *Longer than Deep*. Furthermore at least two spaces of type *Public zone* are slightly out of the perimeter of a Metsovo building (bleu and red rooms).



Furthermore, the whole building synthesis has dimensions that are longer than deep. The final design synthesis is very well adapted to the Metsovo style, while it provides a slightly different composition which it is still characteristic of that style, (Figure 6-38).

6.3.2. Roof Morphology

In this second step we evaluate the performance of the system prototype during the search for the roof(s).morphology applying Metsovo stylistic principles. The input process has the same steps as in earlier experiments.

6.3.2.1. Scene A

We define the appropriate objects of *Scene A*, object properties and some relations which are considered as hard constraints. We select from the previous step a building from *Scene 379* the best design from generation 479 (loop 2).

6.3.2.1.1. Definition of Roof(s)

For the specific building the number of the roofs is two (Table 6-8).

| | |
|-----------------|---|
| Number of Roofs | 2 |
|-----------------|---|

Table 6-8 Number of Roofs

Room Properties

All the spaces have specific placement and specific dimensions as they provided by the earlier run of the system, (Table 6-9).

| Type | Placement | | Dimensions | | |
|--------------|-----------|----------|------------|----------|----------|
| | <i>X</i> | <i>Y</i> | <i>L</i> | <i>W</i> | <i>H</i> |
| Kitchen | 14 | 10,5 | 4,4 | 3 | 3 |
| Bedroom | 6,8 | 9 | 4,4 | 3 | 3 |
| Living room | 10 | 5 | 4 | 6,9 | 3 |
| Dinning room | 14 | 5 | 4,3 | 5,5 | 3 |
| Bathroom | 10 | 12 | 1,6 | 1,5 | 3 |
| Guest room | 5,5 | 6 | 4,5 | 3 | 3 |

Table 6-9 Rooms placement – dimensions

Roof Properties

Roofs dimensions are inherited by the spaces that they cover. Roof length and width are both fixed resulted from the bounding box of the spaces that the roof covers. The height of a roof is variable.

Roof Constrains

As hard constraints is which roof covers which type of spaces of a given building. A roof covers as a minimum one space (Figure 6-39).

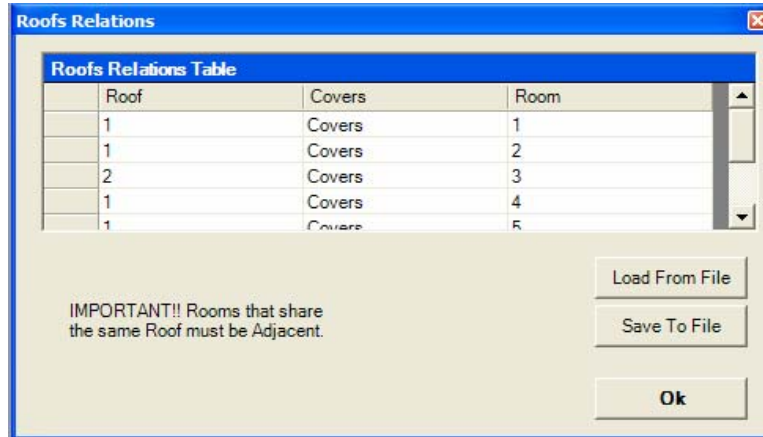


Figure 6-39 Roofs – Rooms Relations

Roof Objective Functions

The style has two objective functions in order that roof(s) become parallel with the greatest dimension of the building, in this case its length.

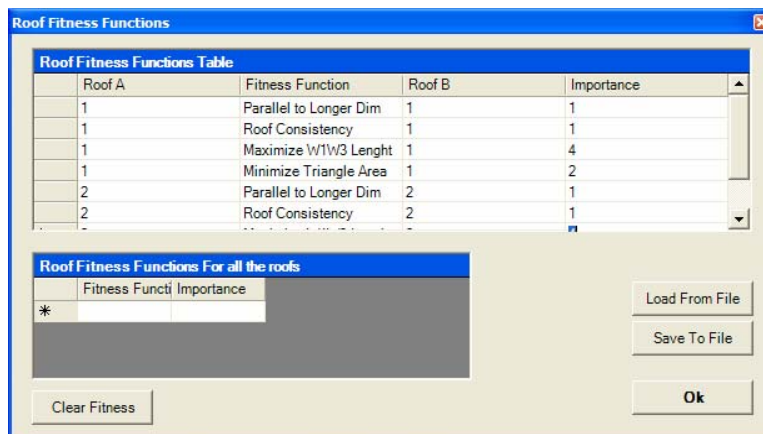


Figure 6-40 Roof Fitness Functions

The fitness is “*Parallel to Longer dimension*” and “*Roof Consistency*”. The “*Maximise W1 – W3*” and “*Minimize Triangle Area*” evaluate the geometry of the roof’s form towards the type of *gable* roof morphology, (Figure 6-40).

6.3.2.1.2. Experiment A

Genetic Algorithm Properties

The GA was run for 800 generations for 5 times, with different initial generation. The population was 100. The mutation rate was 2%. 20 elitist individuals ‘survive’ for 5 generations, (Figure 6-41).

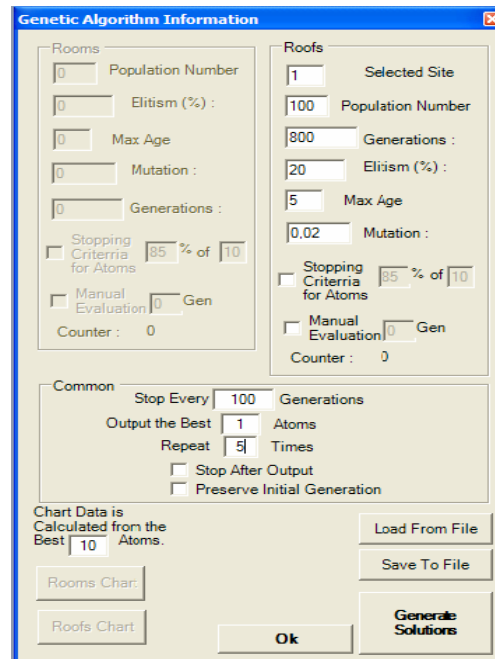
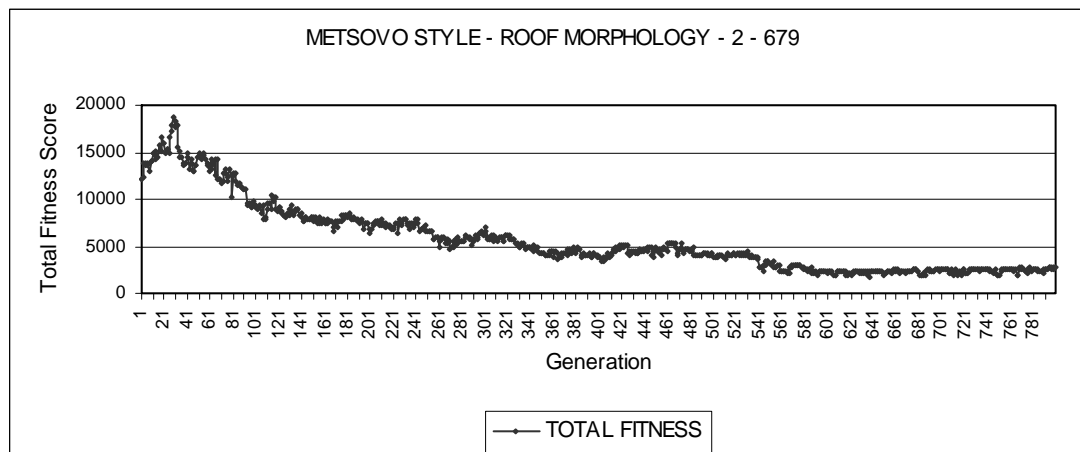


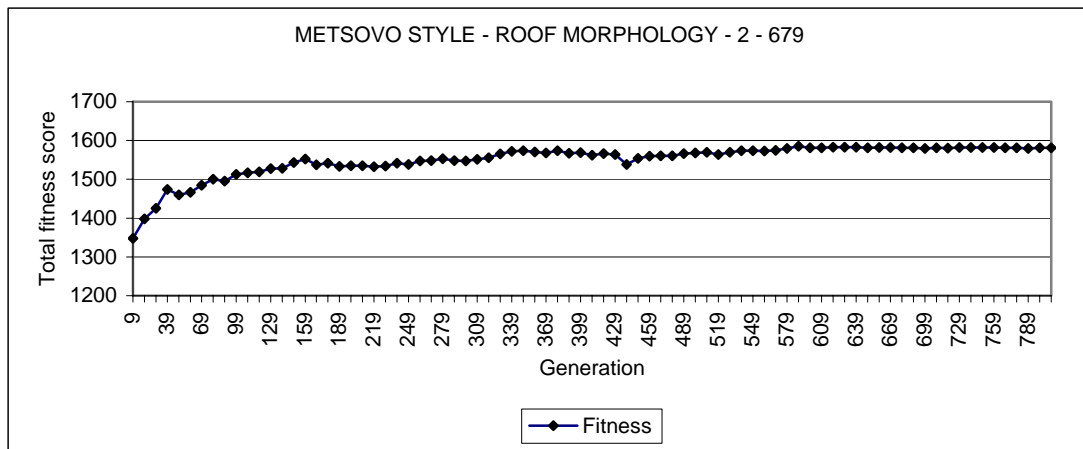
Figure 6-41 Genetic Algorithm Information

Results

The following charts present the sum of all objective functions, for the best individual from all generations. The resulted scores for the best individual are progressive low and remain low till the end of evolution.



In the following chart it is presented the same total score as a sum of normalised objective functions (Chapter 4).



The roof designs gallery (Table 6-10) offers an idea that the MOGA explores a great space of solutions and it is not trapped within local optimal solutions.

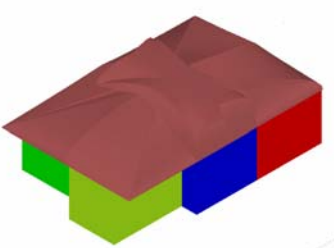
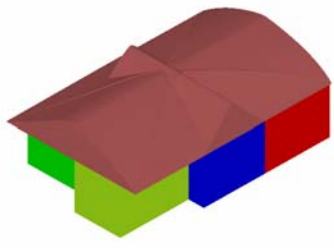
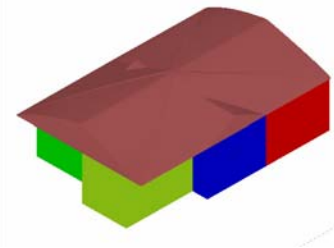
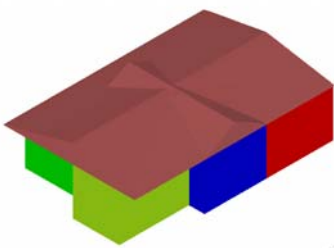
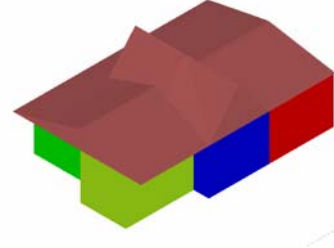
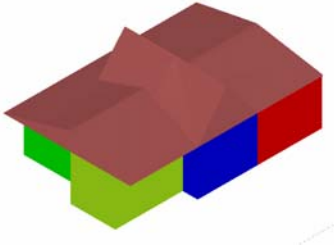
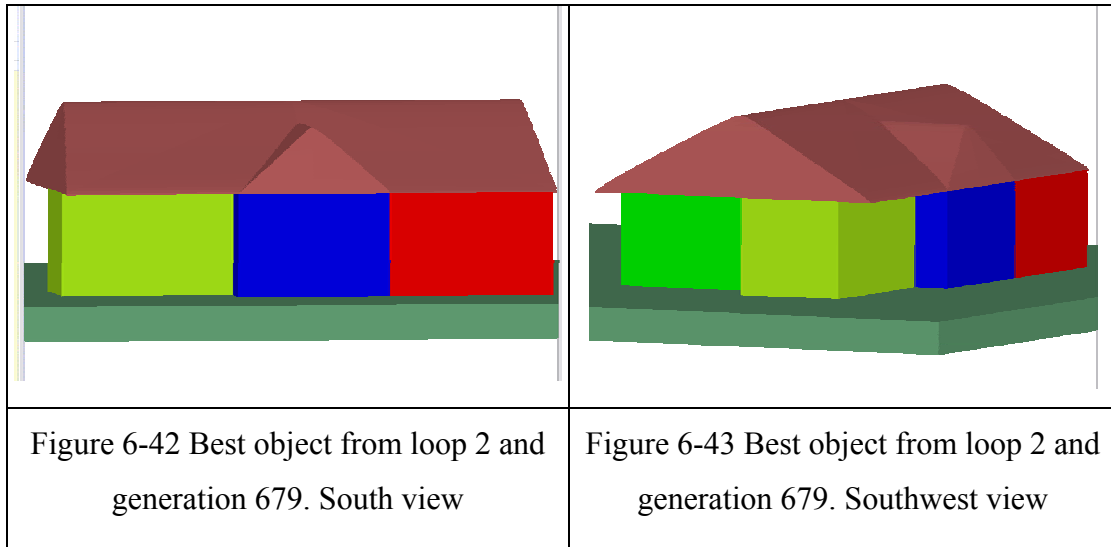
| Repeat time 2 Generation 9 | Repeat time 2 Generation 279 | Repeat time 2 Generation 479 |
|---|---|--|
|  |  |  |
| Repeat time 2 Generation 679 | Repeat time 2 Generation 779 | Repeat time 2 Generation 800 |
|  |  |  |

Table 6-10 Roof designs along generations

Scene solution A

The chosen individual is the best individual of generation number 679 of *Scene 21*, (Figure 6-42). It is clear that the specific roof morphology is very well adapted to the objectives of the Metsovo style, (Figure 6-43).the form of the two roofs is almost exactly a typical Metsovo style roof.



6.3.2.2. Scene B

For this experiment case the selected Scene from the spatial planning is the best individual from generation 829 as evolved during loop 1 from *Scene 367*.

6.3.2.2.1. Definition of Roof(s)

For the specific building requirements the designer has already defines the Number of Roofs for the building. In this design case the number of roofs is two (Table 6-11).

| | |
|-----------------|---|
| Number of Roofs | 2 |
|-----------------|---|

Table 6-11 Number of Roofs

Room Properties

The designer selects a building composition produced from the earlier run of the MOGA, (Table 6-12).

| Type | Placement | | Dimensions | | |
|--------------|-----------|-----|------------|-----|-----|
| | X | Y | L | W | H |
| Kitchen | 14,1 | 10 | 3,9 | 3 | 3 |
| Bedroom | 14,1 | 13 | 3,9 | 2,9 | 3 |
| Living room | 11 | 9,5 | 3,1 | 6,3 | 3 |
| Dinning room | 5,1 | 10 | 5,9 | 3 | 3 |
| Bathroom | 9,1 | 13 | 1,8 | 2,6 | 3 |
| Guest room | 5,5 | 13 | 3,6 | 3 | 3 |

Table 6-12 Rooms placement – dimensions

Roof Properties

All roofs have specific dimensions inherited by the spaces that they will cover. In particular, length and width of a roof are both fixed and are resulted from the bounding box of the spaces that the roof covers. The height of a roof is variable.

Roof Constrains

The designer defines as hard constraints which roof covers which type of space(s) of the given building composition. The relations between the two roofs and the six spaces are presented in the following table (Figure 6-44).

| Roof | Covers | Room |
|------|--------|------|
| 1 | Covers | 1 |
| 1 | Covers | 2 |
| 2 | Covers | 3 |
| 1 | Covers | 4 |
| 1 | Covers | 5 |

IMPORTANT!! Rooms that share the same Roof must be Adjacent.

Figure 6-44 Roofs – Rooms relations

6.3.2.2.2. Roof Objective Functions

The appropriate objective functions are applied for each roof separately. For the first roof the style provides two objective functions in order that ‘presses’ the evolution of roof(s) to become parallel with the greatest dimension of the building, in this case its length. The kind of fitness is *Parallel to Longer dimension* and *Roof Consistency*. The

following *Maximise W1 – W3* and *Minimize Triangle Area* evaluate the geometry of the roof's form for gable morphology, (Figure 6-45).

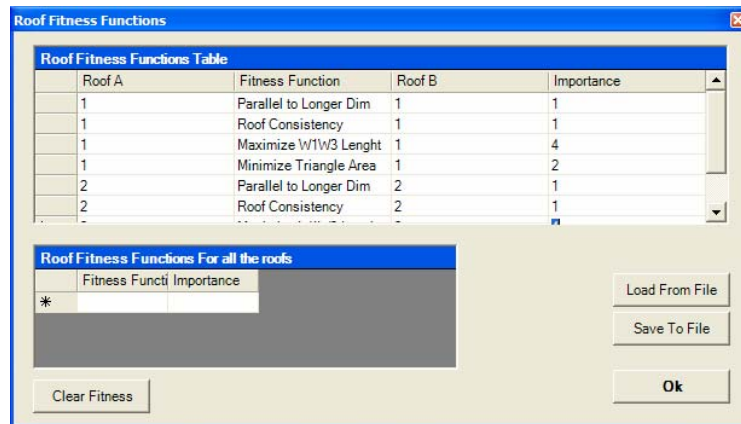


Figure 6-45 Roofs Fitness Functions

6.3.2.2.3. Experiment B

Genetic Algorithm Properties

The GA was run for 5 times, with different initial generation and for 800 generations. The population is 100. The rate of mutation is 2%. Again 20 elitist individuals ‘survive’ for 5 generations before excluding from evolution, (Figure 6-46).

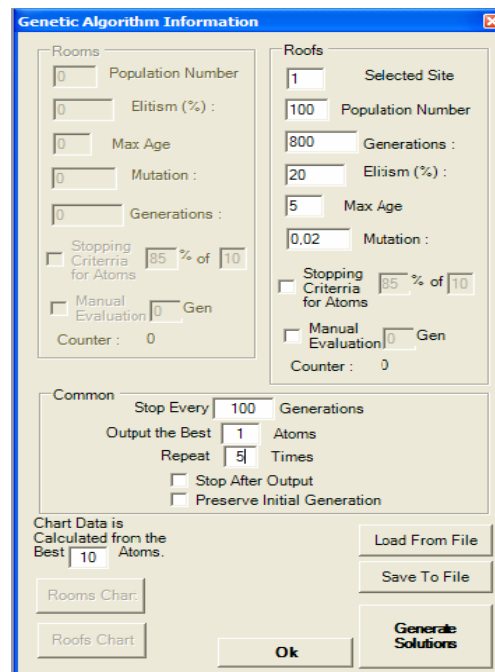
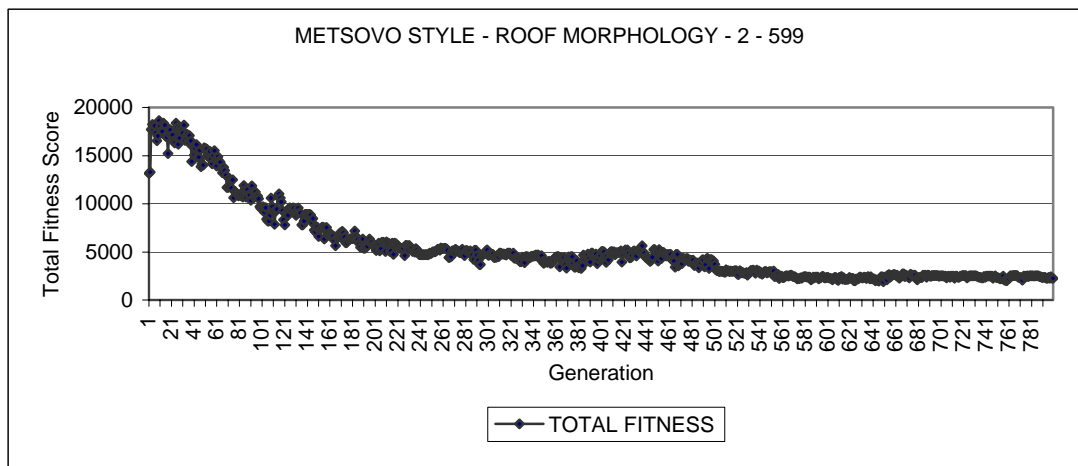


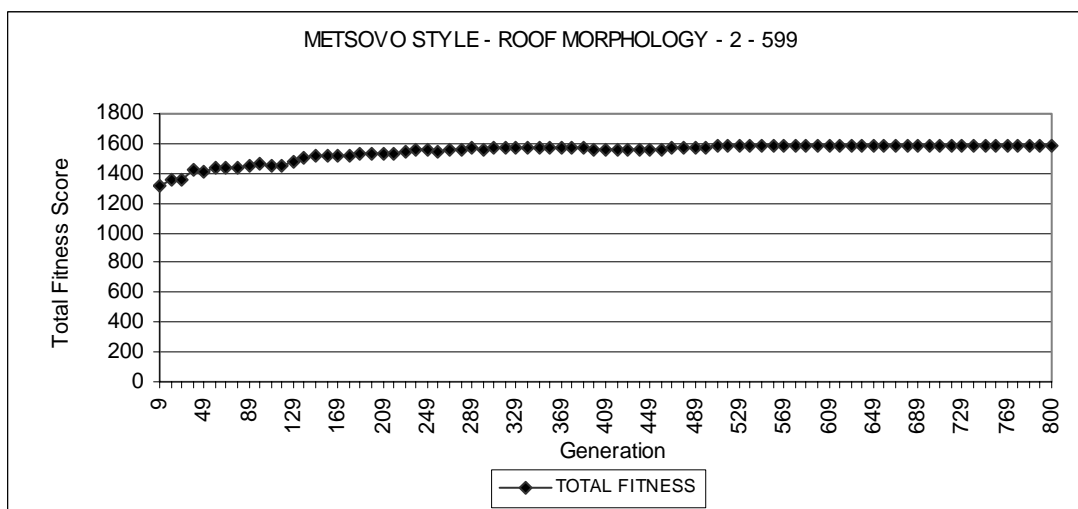
Figure 6-46 Genetic Algorithm Information

Results

The following charts provide useful information for the evolution of roof morphology. We present the sum of all objective functions, for the best individual from all generations. We observe that the resulted scores for the best individual is progressive high and remain high till the end of evolution. However we obtain similar successful results with the earlier experiment. The MOGA converge fast it remains stable till the end of the evolution.

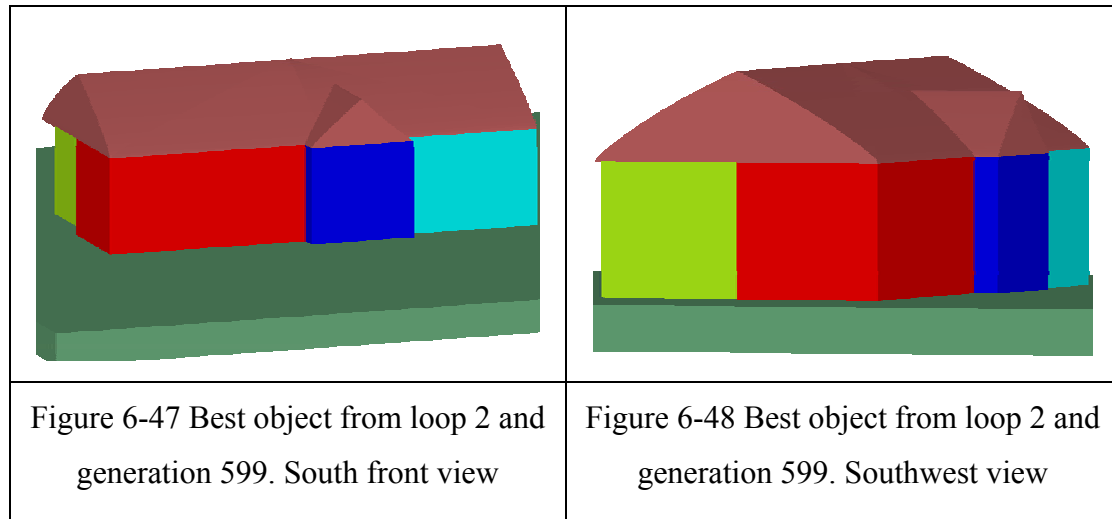


The next chart present the same total scores as a sum of normalised objective functions (chapter 4). According the results we obtain improved individuals representing acceptable roof forms for the soft constraints of the specific style.



Scene solution B

We present as a successful case the best individual of generation number 599 from loop 2 of *Scene 23*, (Figure 6-47). It is clear that the specific roof morphology is very well adapted to the objectives of the Metsovo style, (Figure 6-48).



6.4. Discussion

In order to evaluate the performance of the system we provide a framework for the experiments. The performance of the system is an important criterion for the feasibility of the system.

In this experiment we try to evaluate the performance of the Evolutionary Declarative design system prototype when it is confronting simple building designs with no more than six spaces and at the same time with few hard constraints among them. We enable the spaces to have varied dimensions while they are only constrained by the building brief.

A number objective functions as expression of stylistic principles direct the evolution of the individuals in order for the later to express building designs adapted to an architectural style. The whole process has two distinct steps. Firstly, the designer utilise the Multi-Objective Genetic Algorithm (MOGA) system in order to evolve the spatial planning – composition for given building requirements, under specific principles of an architectural style. Secondly, from the results of the earlier step, in particular from the resulted individual of the last generation, the designer selects one individual in order to evolve the roof morphology of the building. The evaluation

criteria are specific principles of an architectural style. For the purpose of our work we examine two different architectural styles, that of Santorini, and that of Metsovo. They are quite different on both their spatial composition and roof morphology aspects. Both styles were presented in details in an earlier chapter. In the current experiment we set up two design case studies and we introduced it within the evolutionary declarative design system. For the first study a set of evaluation criteria was based on the design principles of architectural style of Santorini. For the second study a set of evaluation criteria based on Metsovo style. From a series of consecutive runs of the system under the description of two alternative settings for the MOGA, for the same Scene we obtain the following results.

The Evolutionary Design system has produced well adaptive building designs according to the stylistic criteria for both styles. Besides, the solutions were without violation of the hard constraints that imposed by the designer description. The multiobjective optimisation method (MOGA) has dealt with seven objective evaluation functions in the case of Santorini, and ten objectives in the case of Metsovo style. We observe that all objective criteria optimised at an appropriate level, given the fact of their relevant degree of importance.

The evolved designs were along the evolution obtain the special characteristics of either Santorini or Metsovo style concerning both the building's spatial planning and the building's roof morphology. Along the consecutive runs of the evolutionary design environment the system provided with well performance, while the MOGA avoids get trapped to local optima. The MOGA has provided the designer of the system with successful designs while it provides variations of the specific architectural style. Especially the later result proves that the selection of the specific stylistic criteria were the appropriate in order to provide the designer with feasible and alternative variations of the same style.

Furthermore, the charts remind us that the scope of the MOGA it is not to obtain *the* optimum solution. The scope it is rather to provide a range of high score solutions well adapted to the objectives of the particular style. Another result was that the specific evolutionary design environment based on MOGA could enable the easy use of stylistic objective criteria.

The MOGA has provided the designer of the system with successful designs while it provides variations of the basic roof morphology of the two specific architectural styles. Especially the later result proves again that the selection of the specific stylistic

criteria were the appropriate in order to provide the designer with feasible and alternative variations for roof morphology of the same style. By the generated solutions is proved that the selection of the specific stylistic criteria were the appropriate in order to provide the designer with feasible and alternative variations of the same style. Another result was that the specific evolutionary design environment based on genetic algorithm could enable the easy use stylistic objective criteria. We consider three aspects of particular interest when assessing the performance of the evolutionary methods: effectiveness, efficiency and robustness. Effectiveness usually refers to the quality of the solutions produced by the method. Efficiency usually refers to how much computation time and memory the method uses. Robustness usually refers to how consistent the method is in producing the same or very similar results over many runs on the same problem instance. The quality of the solutions is observed as very high based on the fact that the generated solutions are similar to original examples of edifices from each style. As it concerns the efficiency aspect of the system, we observe that the system is consume very little computation time and memory. Furthermore the applied MOGA method provides high degree of robustness. In all runs the method generates well-performed solutions. Additionally, it provides the designer with variation within the specific architectural style.

Chapitre 7

Discussion - Conclusions

7.1. Discussion

Dans ce travail nous avons présenté l'hypothèse suivante : la possibilité d'introduire la dimension de style dans la modélisation déclarative afin de faciliter la conception architecturale pendant la phase conceptuelle.

Pour étudier une telle hypothèse, une méthodologie de la phase de conception architecturale utilisant le style a été développée. La méthode est placée dans le cadre de la modélisation déclarative et des algorithmes évolutifs.

L'impact du style est très important lors du processus de conception architectural. Cependant, la modélisation de style est difficile en raison de son caractère qualitatif. Pour cette raison, l'introduction du style dans un système de conception assistée par ordinateur est pénible et compliquée. Les expériences qui ont été accomplies avec le système de prototype développé fournissent la preuve que la représentation de style est possible. En fait, les exemples introduits sont ceux de deux styles architecturaux distincts. La méthode est basée sur des attributs compositionnels, topologiques et géométriques pour représenter le style. A un premier niveau, un concepteur pourrait introduire les propriétés et les relations spécifiques sur la composition et la morphologie d'un bâtiment afin qu'il s'ajuste à un modèle particulier. A un deuxième niveau, le style est formé comme un ensemble de fonctions objectives afin d'être employé dans un algorithme génétique multi-objectifs pour qu'apparaissent des conceptions alternatives adaptées à un style. Les expériences ont été couronnées de succès. En particulier, l'architecte a obtenu un nombre de solutions possibles (synthèse de bâtiments) qui sont bien adaptés à un style préféré. Dans notre système, deux processus semblables mais distincts sont examinés. Le premier concerne le développement de style et le second l'utilisation du style. Les résultats expérimentaux pour le développement de deux modèles ont prouvé que cela était faisable. Une série

d'expériences avec des descriptions de bâtiments différents ont fourni des alternatives de conception adaptées aux deux modèles appliqués.

La modélisation déclarative jusqu'ici n'avait pas traité des domaines tels que la conception architecturale. Ceci se produit parce que cette dernière a un caractère complexe inhérent. La conception architecturale est un ensemble d'activités très exigeantes tandis que certaines d'entre elles se caractérisent par leur haut niveau de créativité. Le processus de conception se caractérise par: sa complexité, son manque de précision et son degré élevé de subjectivité. La synthèse de la structure spatiale tridimensionnelle d'un bâtiment est l'une des tâches ouvertes les plus importantes dans l'architecture. En outre, la recherche de la morphologie d'un bâtiment est une tâche très complexe et pénible. Tout le processus pour une bonne solution est également une recherche de l'information appropriée avec laquelle on procédera à l'évaluation. Ainsi l'application de la modélisation déclarative dans la conception architecturale présente de nombreuses difficultés. La méthodologie pour la conception architecturale est basée sur la modélisation déclarative. Le système développé permet d'aborder avec succès la phase conceptuelle de la conception architecturale. À partir d'une série d'expériences, le système répond à beaucoup de questions de conception architecturale d'un bâtiment.

Afin de tirer profit des capacités de la modélisation déclarative dans la conception architecturale, nous avons abordé les problèmes à partir de trois points de départ prenant en considération les trois phases du cycle conceptuel.

D'abord, dans la phase de description, la description d'une scène est ouverte à plusieurs interprétations. Ce problème est traité par un cadre pour l'introduction de la connaissance de domaine spécifique et de la connaissance architecturale, dans une base de connaissance de type modèleur déclaratif. La *modélisation déclarative par décomposition hiérarchique (DMHD)* fournit une excellente infrastructure pour notre cadre. Le « cadre de Connaissance Déclaratif pour la Modélisation de Bâtiments orienté vers l'Architecture » (*CDMBA*) est présenté dans MultiCAD et offre une typologie d'entités, de propriétés et de relations employées afin d'identifier des rapports sémantiques requis pour définir des modèles de bâtiments. L'application de *CDMBA* permet la création de *Modèles de Bâtiments Déclaratifs Normalisés (MBDN)*. Par conséquent, en introduisant des alternatives de *CDMBA*, faisant appel à la même

connaissance de domaine spécifique, des groupes différents de *MBDNs* peuvent être produits. Chaque groupe contient des scènes de bâtiments exprimées selon la même structure catégorique. Une implémentation du système d'architecture MultiCAD dans un système CAO commerciale et un système de base de données a donné de bons résultats. La description résultante, basée sur un dialogue des modèles déclaratifs, offre beaucoup d'améliorations. En particulier :

- Elle permet de cerner les exigences de l'utilisateur de manière déclarative pendant le processus de la phase de conception.
- Elle facilite le processus de développement de modèles de bâtiments cohérents par l'exploitation de la connaissance architecturale, placée dans le système dans une structure appropriée.
- Elle permet l'édition des paramètres et des contraintes de conception dans n'importe quelle phase du processus déclaratif d'une manière interactive et facile à utiliser.
- Elle favorise le perfectionnement des scènes de bâtiments normalisées.
- Elle facilite le développement de composants d'interface afin de rendre plus aisée ce genre de description.
- Elle fournit des mécanismes de cohérence selon les *CDMBA* appliqués, qui sont enrichis avec des contraintes spécialisées relatives au style architectural et à la typologie des bâtiments.

Deuxièmement, jusqu'ici les modeleurs déclaratifs ont utilisé des techniques *CSP* pour l'exploration de l'espace de solutions et par conséquent, pour la génération des solutions. Ces approches sont intéressantes pour une recherche exhaustive de l'espace de solutions. Avec des inconvénients et des limitations (longue durée de génération et nombreuses solutions de scène), cette approche s'est révélée dans de nombreux cas réussie. Cependant ces approches ne sont pas très appropriées pour la conception assistée (CA). L'évaluation des solutions alternatives concerne généralement de nombreux critères, dans beaucoup de cas complexes et conflictuels entre eux. Dans bien des problèmes de conception architecturale, la satisfaction simultanée de plusieurs objectifs est essentielle. Un architecte pendant la conception de la phase de conception a besoin de rechercher un nombre d'espaces combinatoires impressionnant

d'une manière *non - exhaustive* et *guidée*. Par conséquent, nous proposons un nouveau type de moteur génératif, celui de l'algorithme génétique. La première application d'un algorithme génétique simple fournit avec succès des résultats qui valident son utilisation dans un modèleur déclaratif. La deuxième application d'un algorithme génétique (AG) multi-objectif fournit des résultats prometteurs. En outre, pendant la phase de compréhension de la scène, l'utilisateur a une tâche pénible et dure à accomplir en raison de nombreuses scènes en résultant. L'introduction de l'*Algorithme Génétique Multi-Objectifs (AGMO)* produit un ensemble de solutions déjà évaluées après des critères spécifiques (légères contraintes). Pour cette raison, l'utilisateur a un nombre significativement plus petit de solutions à évaluer.

Troisièmement, habituellement la modélisation déclarative a traité des contraintes rigides, qui doivent toutes être satisfaites. La phase de conception *détermine le principe* d'une solution. Un nombre de problèmes de base de la phase de conception sont les suivants. Dans un problème donné, il existe des contraintes et des objectifs mais leur distinction et leur classification est très souvent très imprécise. Dans beaucoup de cas, l'examen et la compréhension du problème aboutissent sur un déplacement des objectifs aux contraintes ou vice versa. La nature des contraintes pourrait être rigide ou légère. L'expansion continue des problèmes exige de certains d'entre eux de changer de caractère rigide à léger et vice versa. Il est toujours possible d'éliminer certaines des contraintes pendant le processus. Cependant, dans la conception architecturale conceptuelle il y a des contraintes rigides et des contraintes légères. Afin d'exploiter les capacités de la modélisation déclarative dans la conception architecturale, nous fournissons une telle séparation de contraintes. Puis, nous introduisons des contraintes légères en tant que fonctions objectives dans un algorithme génétique multi-objectif. Dans une série d'expériences, nous observons que l'algorithme génétique multi-objectif n'a pas violé des contraintes rigides tandis qu'il produit des résultats bien adaptés aux contraintes légères.

7.2. Conclusion

L'hypothèse principale de cette thèse est la possibilité de l'introduction de la dimension du style dans le cadre de la modélisation déclarative afin de faciliter la conception architecturale pendant la phase conceptuelle, en tant que réponse positive.

Une méthodologie de la phase de conception architecturale utilisant le style et basée sur le cadre de modélisation déclarative et d'algorithmes évolutifs a un résultat important. Le résultat est le système prototype de Conception Déclarative Évolutionnaire - MultiCAD (*CDE-MultiCAD*) pour l'aide de la conception architecturale conceptuelle.

La satisfaction de notre hypothèse au-delà du résultat apparent du système prototype est bénéfique à un éventail de phénomènes relatif à la conception.

Le système prototype de conception résultant prend en compte les aspects esthétiques et particulièrement, la dimension du style. Le concepteur peut concevoir des bâtiments adaptés à un modèle architectural particulier.

La méthode résultante pourrait quantifier avec succès des critères qualitatifs esthétiques d'un bâtiment. Un modèle de style a pu permettre la modélisation de style. La quantification du style a pu fournir aux architectes plus de critères de décision pendant la génération des solutions. Les architectes ont un outil opérationnel pour modeler leurs préférences esthétiques et les faire évoluer. En général, de tels outils pourraient fournir la possibilité aux concepteurs – architectes de distiller des critères complexes et qualitatifs. De plus, c'est un système prometteur de Prise de Décision en Conception basé sur des critères esthétiques. Il est évident qu'un tel système se tourne vers un outil de conception esthétique assisté par ordinateur.

7.2.1. Modélisation déclarative et phase de conception architecturale

Le cycle conceptuel déclaratif fournit un outil efficace pour la phase de conception architecturale. L'introduction de la connaissance architecturale et du style parallèlement à l'adoption des algorithmes évolutionnaires améliorent ses capacités. L'applicabilité des systèmes de modélisation déclaratifs dans la conception architecturale est grande. L'introduction du paradigme déclaratif évolutionnaire dans la Conception Architecturale Assistée par Ordinateur (*CAAO*) s'attaque à la phase

exigeante du processus de conception architecturale du début. Un tel outil accroîtra les capacités du concepteur à produire des concepts de conception nouveaux.

7.2.2. Esthétique et intelligence artificielle

La dimension esthétique sous forme de principes stylistiques peut être mise en œuvre par des techniques d'intelligence artificielle. Des ensembles de critères stylistiques (esthétiques) ont été employés tant pour la génération que pour l'évaluation des produits de conception et facilitent les décisions finales du concepteur. Une telle méthodologie amplifie la créativité dans le processus de conception. Les architectes et les chercheurs sont équipés d'un outil pour réaliser une compréhension quantitative des concepts esthétiques impliqués dans le processus de conception.

La méthode proposée a pu avoir un impact important sur des études sur la formation, la représentation et l'utilisation du style dans la conception architecturale et au-delà. D'ailleurs, cela pourrait constituer un outil prometteur pour les études sur la morphologie de construction. La morphologie, l'étude du modèle et de la forme, est cruciale à la conception parce qu'elle constitue une partie essentielle de son corpus de connaissance logique.

En général, l'application actuelle peut contribuer à la recherche et à la conception architecturale et en plus, pourrait avoir une influence rétroactive sur la théorie de conception.

Cependant, l'environnement de conception pour l'application de nos idées était la phase de conception des constructions, beaucoup de méthodes et de techniques pourraient être appliquées à d'autres secteurs de conception.

CDE-MultiCAD avec les adaptations appropriées pourrait être applicable à un type plus étendu de la phase de conception de produit, comme la conception intérieure, la conception de meubles et le design industriel.

7.3. Travaux futurs - Questions ouvertes

7.3.1. Expérimentation du système de prototype

L'essai du système a été réalisé dans trois directions. Premièrement, la représentation de plus de styles architecturaux, tant historique que de grands architectes.

Deuxièmement, en augmentant le niveau de détail sur la représentation de style permettrait l'étude analytique et détaillée du style. Les éléments stylistiques ont pu être encore enrichis avec l'addition des objets de construction comme des colonnes, des fenêtres, des portes, etc. Troisièmement, l'évaluation du système de prototype avec les études de cas de constructions plus complexes et de plus grandes échelles.

7.3.2. Extensibilité du processus d'évaluation

Une extension de l'environnement d'évaluation pourrait être le suivant: Le système prototype pourrait être fourni à un certain nombre de critiques (des d'experts en architecture -conception). Les évaluations pourraient avoir de nombreux objectifs. Examiner la validité du système, définition d'un niveau approprié de détail, espérances de rendement.

7.3.3. Extensibilité à d'autres domaines de conception

Le concept du style existe dans beaucoup de domaines de conception : dans la conception de produits dans la conception intérieure et la conception de meubles. L'étude de ces concepts pourrait permettre l'utilisation plus large du système de prototype dans nouveaux domaines de conception.

7.3.4. Application - Utilisation en tant qu'outil d'enseignement de conception

Le système avec des ajustements appropriés pourrait fournir un outil explicatif pour enseigner la formation et l'évolution des concepts stylistiques.

Index of Abbreviation

| | |
|----------|---|
| AEC | Architecture, Engineering and Construction |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| BCCM | Building Construction Core Model |
| CAAD | Computer-Aided Architectural Design |
| CAD | Computer-Aided Design or Drawing |
| CAPD | Computer Aided Product Design |
| CAS | Computer Aided Styling |
| CFI | CAD Framework Initiative |
| CIM | Computer Integrated Manufacturing |
| CIMsteel | Computer Integrated Manufacturing System for STEEL work. |
| CMF | Conceptual Modelling Framework |
| COMBINE | COMputer Models for the Building INdustry in Europe |
| CSG | Constructive Solid Geometry |
| CSP | Constraint Satisfaction Problem |
| DFM | Design for Manufacturing |
| DKABM | Declarative Knowledge for Architecture Building Modelling |
| DM | Declarative Modelling |
| DMHD | Declarative Modelling by Hierarchical Decomposition |
| EAs | Evolutionary Algorithms |
| EDM | Engineering Data Model |
| EE-R | Extended Entity-Relationship |
| EXPRESS | The language used in STEP |
| FBM | Feature-Based Modelling |
| GARM | General AEC Reference Model |
| GAs | Genetic Algorithms |
| GUI | Graphical User Interface |
| HVAC | Heating Ventilation Air Conditioning |
| IAI | International Alliance for Interoperability |
| IBDS | Integrated Building Design Systems |

| | |
|-------|---|
| ICAM | Integrated Computer Aided Manufacturing |
| IDEF | ICAM DEFinition |
| IDM | Integrated Data Model |
| IDR | Internal Declarative Representation |
| IFC | Industry Foundation Classes |
| IMD | Internal Model Description |
| ISO | International Standards Organization |
| IT | Information technology |
| ML | Machine Learning |
| MOGA | Multi-Objective Genetic Algorithm |
| MOO | Multi-Objective Optimisation |
| NDBM | Normalized Declarative Building Model |
| NIAM | Nijssen's Information Analysis Method |
| PDM | Product Data Model |
| STEP | STandard for the Exchange of Product model data |
| SUS | Stochastic Universal Sampling |
| UML | Unified Modelling Language |
| V.T.T | Technical Research Centre of Finland. |
| VRML | Virtual Reality Modelling Language |

Bibliography

- [Ackerman 63] Ackerman J. S., "Style", in Art and Archaeology Eds J S Ackerman, R Carpenter (Prentice-Hall, Englewood Cliffs, NJ) pp 174-186, 1963
- [Akin 86] Akin O., "Psychology of Architectural Design", Pion, London, 1986
- [Akin 96] Akin O., Akin C., "Frames of reference in architectural design: analysing the hyperacclamation (A-h-a!) Design Studies, 17, 341-361, 1996
- [Amor 95] Amor R., Augenbroe G., Hosking F., Rombouts W., Grundy J., "Directions in modelling Environments", Automation in Construction, 4(3) 173-187, 1995
- [Amor 98] Amor R., "A Survey and Analysis of Integrated Project Databases", CIBW78, Sweden, 1998
- [Augenbroe 95] Augenbroe G. "COMBINE 2 Final Report", Delft: Delft University of Technology, 1995
- [Bäck 97] Bäck Th., Fogel D. B., Michalewicz Z., (eds.), "Handbook of Evolutionary Computation", Oxford University Press, New York, 1997
- [Banham 80] Banham, R., "Theory and Design in the First Machine Age", MIT Press, Cambridge, MA, 1980
- [Bardis 04] Bardis G., Miaoulis G., Plemenos D., "An Intelligent User Profile Module for Solution Selection Support in the Context of the MultiCAD Project". 3IA'04 International Conference, Limoges, France, 2004
- [Bardis 05a] Bardis G., Miaoulis G., Plemenos D. "Learning User Preferences at the Declarative and Geometric Description Level", 3IA'05 International Conference, Limoges, France, 2005
- [Bardis 05b] Bardis G., Miaoulis G., Plemenos D., "Intelligent Solution Support Based on Alternative User Profiles", International Conference of Enterprise Information Systems (ICEIS),

- Miami, USA, 2005
- [Bentley 01] Bentley P., (eds.) “Creativity and Design”, Morgan Kaufman press, 2001
- [Bentley 96] Bentley P., “Generic Evolutionary Design of Solid Objects Using a Genetic Algorithm” PhD thesis, University of Huddersfield, 1996
- [Bentley 99] Bentley P., (eds.) “Evolutionary Design by Computers” Morgan Kaufmann Publishers Inc, San Francisco, 1999
- [Bidarra 99] Bidarra R. “Validity maintenance in semantic feature modelling”, PhD thesis, Delft University of Technology, The Netherlands, 1999
- [Bidarra 00] Bidarra R., Bronsvort W. F., “Semantic feature modelling”, *Computer-Aided Design*, 32 (3) p. 201–225, 2000
- [Björk 92a] Björk B.-C., “A conceptual model of spaces, space boundaries and enclosing structures”, *Automation in Construction*, 1(3) 193-214, 1992
- [Björk 92b] Björk B.-C., “A Unified approach for Modelling Construction Information” *Building and Environment*, 27(2) 173-194, 1992
- [Björk 94] Björk B.-C., “RATAS project—Developing an infrastructure for computer-integrated construction.”, *Journal of Computing in Civil Engineering*, ASCE, 8(4) 401–419, 1994
- [Björk 95] Björk B.-C., “Requirements and Information structures for building product data models”, Dissertation for the degree of Doctor of Technology, Helsinki University of Technology, VTT, Espoo, Finland, 1995
- [Bonnetfoi 00] Bonnetfoi P. F., Plemenos D., “Constraint satisfaction techniques for declarative scene modelling by hierarchical decomposition” 3IA’00 International Conference, Limoges (France), 2000
- [Bonnetfoi 99] Bonnetfoi P. F., “Techniques de satisfaction de contraintes pour la modélisation déclarative. Application à la génération concurrente de scènes”. PhD thesis, Limoges, France, 1999

- [Breemen 98] van Bremen E. J. J., Sudijone S., Horvath I., "A contribution to finding the relationship between Shape Characteristics and Aesthetic Appreciation of Selected Product", Proc. of the 12th International Conference on Engineering Design ICED99, 24-26 August 1999, Munich Lindemann, U., Birkhofer H., Meerkamm H., Vajna, S., (Eds.), Munich 1999, pp. 1765-176, 1999
- [Broadbent 90] Broadbent G., "Emerging Concepts in Urban Space Design", Chapman & Hall, London, 1990
- [Bronsvoort 93] Bronsvoort W. F., Jansen F. W., "Feature modelling and conversion – Key concepts to concurrent engineering", Computers in Industry, 21 (1): 61-86, 1993
- [Bronsvoort 01] Bronsvoort W., F., van den Berg E., Bidarra R., Noort A., "Essential Developments in Feature Modelling", CAD/Graphics'2001 August 22-24 International Academic Publishers, Kunming, 2001
- [Brown 00] Brown, D. C., "Artificial Intelligence Views of Conceptual Design." Keynote Speech on the Third International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID&CD'00. Nov., 2000. Hong Kong, 2000
- [Buchwald 99] Buchwald H., "Form, Style and Meaning in Byzantine Church Architecture" Variorum Collected Studies, 1999
- [Caldas 01] Caldas L. G., "An Evolution-Based Generative Design System: Using Adaptation to Shape Architectural Form" PhD thesis Massachusetts Institute of Technology Massachusetts, 2001
- [Carrara 94] Carrara G., Kalay Y. E., Novembri G., "Knowledge - Based Computational Support for Architectural Design", Carrara G., Kalay Y. E., (eds.), "Knowledge-Based Computer-Aided Architectural Design", Elsevier Science Publishers B.V., Amsterdam, NL, 1994
- [Champiaux 98a] Champiaux L., "Introduction de techniques d'apprentissage

- en modélisation déclarative”. PhD thesis, Ecole des Mines de Nantes, Juin 1998
- [Champiaux 98b] Champiaux L., “Classification: a basis for understanding tools in declarative Modelling”. *Computer Networks and ISDN Systems* 30 1841–1852, 1998
- [Chan 00] Chan C. S., “Can style be measured?” *Design Studies* 21 207-274. 2000
- [Chan 01] Chan, C. S., “An examination of the forces that generate a style” *Design Studies* 22 319–346, 2001
- [Chan 92] Chan C. S., “Exploring individual style in Design” in *Environment and Planning B: Planning and Design* 19, 503-523, 1992
- [Chan 93] Chan C. S., “How an individual style is generated” in *Environment and Planning B: Planning and Design* 20, 391-423, 1993
- [Chan 94] Chan C. S., “Operational definition of Style” in *Environment and Planning B: Planning and Design* 21, 223-246, 1994
- [Chen 97] Chen, K., Owen C., “Form language and style description”, *Design Studies* Vol 18 pp 249–274, 1997
- [Ching 96] Ching F. D. K., “Architecture: Form, Space & Order”, 2nd Edition, John Wiley and Sons, 1996
- [Chiou 95] Chiou S.-C., Krishnamurti R., “The grammar of Taiwanese traditional vernacular dwellings” *Environment and Planning B: Planning and Design* 22 p689-720, 1995
- [Chiou 96] Chiou S.-C., Krishnamurti R., “Example Taiwanese traditional houses” *Environment and Planning B: Planning and Design* 23 p191-261, 1996
- [Chun 97] Chun H.-W., Lai E. M.-K., “Intelligent Critic System for Architectural Design”, *IEEE Transactions of Knowledge and Data Engineering*, 9(4) 625-639, 1997
- [Clark 96] Clark, R. H., Pause M., “Precedents in Architecture”, Second Edition, Van Nostrand Reinhold, New York. 1996
- [Coates 99] Coates P., Makris D., “Genetic Programming and Spatial

- Morphogenesis". AISB'99 Symposium on Creative Evolutionary Systems. University of Edinburgh, Edinburgh, UK, 1999
- [Coello 00a] Coello C. A. C., "Use of self-adaptive penalty approach for engineering optimization problems", *Computers in Industry* 41, pp. 113-127, 2000
- [Coello 00b] Coello C. A. C., "Handling Preferences in Evolutionary Multiobjective Optimization : A Survey", 2000 Congress on Evolutionary Computation, volume 1, pages 30-37, Piscataway, New Jersey, 2000
- [Coello 02] Coello, C. A. C., van Veldhuizen D.A., Lamont, G. B., "Evolutionary Algorithms for Solving Multi-Objective Problems", Kluwer Academic/Plenum Publishers, New York, 576 pp., 2002
- [Coello 96] Coello C. A. C., "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design", PhD thesis, Department of Computer Science, Tulane University, Mexique (1996).
- [Coello 99] Coello C. A. C., "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems", *An International Journal*, 1(3), pp. 269-308, 1999
- [Colin 97] Colin C., Desmontils E., Martin J. Y., Mounier J. P., "Working modes with declarative modeller", *Compugraphics'97*, 1997
- [Collette 02] Collette Y., Siarry P., "Optimisation multiobjectif" Eyrolles ed., Paris, pp 322, 2002
- [Coyne 90] Coyne, R. D., Rosenman M. A., Radford A. D., Balachandran M., Gero J. S., "Knowledge-Based Design Systems", Addison Wesley, Reading, MA, USA, 1990
- [Cross 84] Cross N., "Developments in Design Methodology" , Chichester Wiley & Sons Ltd. 1984
- [Cross 96] Cross N., Christiaans H., Dorst K., (eds.), "Analyzing Design

- activity”, J. Wiley & Sons, 1996
- [Cvetkovic 00] Cvetkovic D., “Evolutionary Multi-Objective Decision Support Systems for Conceptual Design”, PhD Thesis, School of Computing, Faculty of Technology, University of Plymouth, 2000
- [Cvetkovic 98] Cvetkovic, D., Parmee, I. C., Webb, E., “Multi-objective optimisation and preliminary airframe design”, in [Parmee 98], pp. 255–267, 1998
- [Dasgupta 97] Dasgupta, D., Michalewic, Z. (eds.), “Evolutionary Algorithms in Engineering Application”, Berlin, Germany: Springer, 1997
- [Davis 91] Davis, L., “Handbook of Genetic Algorithms”, New York: Van Nostrand Reinhold, 1991
- [Dawkins 86] Dawkins, R., “The Blind Watchmaker” Longman Scientific & Technical Publication, 1986
- [Deb 91] Deb K., “Binary and Floating Point Function Optimization using Messy Genetic Algorithms” Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 91004, 1991
- [Deiman 93] Deiman, E. P., Platt, H. T., “Cost Information in Succeeding Stages of the Design Process in Advanced Technologies”, Elsevier Science Publisher, 1993
- [Ding 01] Ding L., Gero J., “The emergence of the representation of style in design”, Environment and Planning B: Planning and Design, volume 28, pages 707-731, 2001
- [Dym 94] Dym, C. L., “Engineering Design: A Synthesis of Views”, Cambridge University Press, 1994
- [Eastman 94] Eastman C. M., Fereshetian N., “Information models for use in product design: a comparison” Computer-Aided Design Vol. 26, No 7, pp 551-572, 1994
- [Eastman 95] Eastman C. M., Siabiris A., “A Generic Building Product Model Incorporating Building Model Information”, Automation in Construction, 3(1) 283-304, 1995
- [Eastman 99] Eastman C. M., “Building Product Models: computer

- environments supporting design and construction”, CRC Press, Florida, USA, 1999
- [Ekholm 98] Ekholm A., Fridqvist S. A “Dynamic Information System for Design Applied to the Construction Context”. In: *The Life-cycle of Construction IT Innovations* (Eds. Björk, B-C. and Jägbeck, A.), proceedings from CIB W78 workshop, 3-5 June pp. 219-232, Stockholm, Sweden 1998
- [Evbuomwan 96] Evbuomwan, N., Sivaloganathan, S., Jebb, A. (1996) “A Survey of Design Philosophies, Models, Methods and Systems”, in *Proceedings of the Institution of Mechanical Engineers*, London, Part B: Journal of Engineering Manufacture, Vol. 210, pp. 301-319. 1996
- [Ezzegaf 00] Ezzegaf M., Mareschal B., “Utilisation d'échelles qualitatives dans les méthodes PROMETHEE”, Université Libre de Bruxelles, Institut de Statistique et de Recherche Opérationnelle, 2000
- [Fenves 94] Fenves S., Flemming U., Hendrickson C., Maher M. L., Quadrel R., Terk M., Woodbury R., “Concurrent Computer-Aided Integrated Building Design”, Prentice-Hall, Englewood Cliffs, NJ, USA 1994
- [Flemming 95a] Flemming U, Woodbury R., “Software environment to support early phases in building design: overview”, *Journal of Architectural Engineering*, (4) 1 147–152, 1995
- [Flemming 95b] Flemming U., Chien S.-F., “Schematic layout design in SEED environment”, *Journal of Architectural Engineering* 4 (1) 162–169, 1995
- [Fonseca 95] Fonseca C.S., Fleming P. J., “An Overview of Evolutionary Algorithms in Multiobjective Optimization”, *Evolutionary Computation*, vol. 3 (1), pp. 1-16, 1995
- [Frazer 01] Frazer J. H., “Creative Design and the Generative Evolutionary Paradigm” in [Bentley 01], 2001
- [Frazer 95] Frazer J. H., “An Evolutionary Architecture”, *Architectural Association Publications*, London, 1995

- [Frazer 99] Frazer J., Tang M. X., Sun J., “Towards a Generative System for Intelligent Design support”. Proceedings of the Fourth Conference on Computer Aided Architecture Design Research in Asia. May, 1999
- [Fribault 03] Fribault P., “Modélisation Declarative d’Espaces Habitable”, (in French), PhD thesis, University of Limoges, France 2003
- [Fridqvist 00] Fridqvist, S. “Property-Oriented Information Systems for Design, prototypes for the BAS·CAAD-System”, PhD thesis, Lund University, Sweden, 2000
- [Galle 95] Galle P., “Towards Integrated, ‘Intelligent’, and Compliant Computer Modelling of Buildings”, Automation in Construction, 4(3) 189-211, 1995
- [Gero, 00] Gero J., Kazakov V., “Adaptive enlargement of state spaces in evolutionary designing”. Artificial Intelligence in Engineering Design, and Manufacturing 00;14:31–8 2000
- [Giannini 03] Giannini F., Monti M., “Design intent-oriented modelling tools for aesthetic design”, WSCG’2003 conference, Plzen, Czech Republic, 2003
- [Gielingh 88] Gielingh W., “General AEC Reference Model (GARM), in Proceedings CIB Conference on the Conceptual Modelling of Buildings”, Ed. P. Christensson, Lund, Sweden, 165-178, 1988
- [Goldberg 89] Goldberg, D. E., “Genetic Algorithms in Search, Optimization, and Machine Learning” Addison-Wesley Publishing Corporation, Inc. 1989
- [Goldberg 91] Goldberg D. E., “Genetic algorithms as a computational theory of conceptual design”. Appl Artif Intell Engng VI:3–16. 1991
- [Goldberg 92] Goldberg, D. E. et al. (1992a). Genetic Algorithms, Noise, and the Sizing of Populations. Complex Systems 6, 333-62.
- [Goldberg 98] Goldberg D. E., “The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World” Department of General Engineering IlliGAL Report No.

- 98004, February 1998
- [Gombrich 60] Gombrich E. H., 'Art and Illusion' Pantheon, New York 1960
- [Grefenstette 97] Grefenstette J., "Rank-Based Selection". In Bäck T., Fogel D. B., Michalewicz Z., (eds.), Handbook of Evolutionary Computation (pp. C2.4:1–C2.4:6). Bristol and Oxford: IOP Publishing Ltd. and Oxford University Press, 1997
- [Grierson 97] Grierson, D. E., "Conceptual Design Using a Genetic Algorithm", Proceedings of the 15th Structures Congress "Building to Last", Part 2 (of 2), Apr 13-16, v2, Portland, OR, USA, Sponsored by: ASCE, p. 798-802, 1997
- [Hendricx 00] Hendricx A., "A Core Object Model for Architectural Design", PhD thesis, K.U.Leuven University, Belgium, 2000
- [Holland 92] Holland, J. H. Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, 1992
- [Horn 94] Horn J., et. al., "Implicit Niching in a Learning Classifier System: Nature's Way", Illinois Genetic Algorithms Laboratory (IlligAL), report no. 94001, 1994
- [Horn 97] Horn J., "Multicriteria Decision Making and Evolutionary Computation", In. Bäck Th., Fogel D. B., Michalewicz Z. (eds.), "The Handbook of Evolutionary Computation", Institute of Physics Publishing, Bristol, UK, 1997
- [Hudson 95] Hudson, M. G., Parmee, I. C., "The Application of Genetic Algorithms to Conceptual Design", Proceedings of the Lancaster International Workshop on Engineering Design, AI System Support for Conceptual Design, Charlotte Mason College, Ambleside, p. 17-36. 1995
- [IFC 01] International Alliance for Interoperability, "IFC Model Implementation Guide", Modelling Support Group, Version 1.0, 2001
- [ISO 92] ISO TC184/SC4/WG4 N34 P5, "Guidelines for the Development and Approval of STEP Application Protocols", 1992

- [ISO 93] ISO WD 10303-Part 22, "Standard Data Access Interface", 1993
- [ISO 96] ISO TC184/SC4/WG1, "Part 106 Draft T100 Building Core Construction Model", 1996
- [ISO 96] ISO DIS 10303 Part 225, "Building Elements Using Explicit Shape Representation ", 1996
- [Jo 96] Jo, J. H., Gero, J. S., "Space Layout Planning Using an Evolutionary Approach", Artificial Intelligence in Engineering, Elsevier Science Limited, 1996
- [Joedicke 85] Joedicke J., "Space and form in architecture", Kramer Verlag, Stuttgart, Germany, 1985
- [Kalay 98] Kalay Y.E., "P3: Computational environment to support design collaboration", Automation in Construction, 8(1) 37-48, 1998
- [Kalay 99] Kalay E.Y., "The Future of CAAD: From Computer-Aided Design to Computer-Aided Collaboration", Proceedings from the 10th Computer Aided Architectural Design Conference, Atlanta, USA, 1999
- [Kanal 88] Kanal, L.V. Cumar. (eds.) Search in Artificial Intelligence. Springer-Verlag, 1988
- [Khajehpour 01] Khajehpour S. "Optimal Conceptual Design of High-Rise Office Buildings" Ph D thesis University of Waterloo, Waterloo, Ontario, Canada, 2001
- [Krier 88] Krier, R., "Architectural Composition", Rizzoli, New York 1988
- [Kwaiter 98] Kwaiter G., "Modélisation déclarative de scènes: étude et réalisation de solveurs de contraintes", PhD. Thesis, Institute of Research in Data processing of Toulouse, France 1998
- [Lawson 90] Lawson, B., "How Designers Think", Butterworth Architecture, London, 1990
- [Lawson 94] Lawson B., "Design in Mind", Butterworth Architecture, London,UK, 1994
- [Levine 94] Levine D., "A Parallel Genetic Algorithm for the Set

- Partitioning Problem”, PhD. Thesis, Argonne National Laboratory, Illinois, USA, 1994
- [Lucas 90] Lucas M., Martin D., Martin P., Plemenos D., “The ExploFormes project: some steps towards declarative modelling of forms”, (in french). Journées AFCET-GROPLAN, Strasbourg (France), BIGRE, no 67, January 1990, pp 35 - 49) 1989
- [Lucas 95] Lucas M., Desmontils, E., “Les Modeleurs Declaratifs Declarative modellers”, Rev. Int. CFAO Inf. Graph. 10 6, 559–585, 1995
- [Makris 01] Makris D., “A Study of Form Generation in Architecture”, Research report MSI 01-03 Université de Limoges Octobre 2001
- [Makris 03] Makris D., Ravani I., Miaoulis G., Skourlas C., Fribault P., Plemenos D., “Towards a domain-specific knowledge intelligent information system for Computer-Aided Architectural Design” 3IA’03, Limoges, France, 2003
- [Makris 99] Makris, D. “Evolutionary Design ENvironments (EDEN)”. MSc thesis. School of Architecture, University of East London, London, UK 1999
- [Martin 88] Martin P., Martin D., “An expert system for polyhedra modelling”, Eurographics’88, Nice, France, 1988.
- [Mathews 95] Mathews, J., Rafiq, M. Y., Adaptive Search to Assist in Conceptual Design of Concrete Buildings, Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering, Civil-Comp Press, Edinburgh, UK 1995
- [Meiss 90] von Meiss P., “Elements of Architecture – From Form to Place”, E&FN Spon, London, 1990
- [Miaoulis 00] Miaoulis G., Plemenos D., Skourlas C., “MultiCAD Database: Toward a unified data and knowledge representation for database scene modelling”, 3IA’00 conference, Limoges, France, 2000

- [Miaoulis 02] Miaoulis G., “Contribution à l'étude des Systèmes d'Information Multimédia et Intelligent dédiés à la Conception Déclarative Assistée par l'Ordinateur – Le projet MultiCAD” (in French), Professorial dissertation, University of Limoges, France 2002.
- [Miaoulis 96] Miaoulis G., Plemenos D., “Propositions pour un système d'information multimédia intelligent dédié à la CAO – Le projet MultiCAD”, Rapport de recherche MSI 96-03, Université de Limoges, France, 1996
- [Michalewicz 95] Michalewicz Z., Nazhiyath G., “Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints”, In Fogel D. B., (Ed.), Proceedings of the Second IEEE International Conference on Evolutionary Computation, pp. 647--651. IEEE Press 1995
- [Michalewicz 96] Michalewicz Z., “Genetic Algorithms + Data Structures=Evolution Programs”, Springer Verlag, 387 pp 1996.
- [Michelis 01] Michelis P., “Aesthetic studies” Vol. 1-2 P. E. Michelis Foundation, Athens, 2001
- [Michelis 02] Michelis P., “Architecture as art” P. E. Michelis Foundation Athens, 2002
- [Michelis 81] Michelis P., “The Greek Traditional House” (in Greek), EMP Press, Athens, Greece, 1981
- [Mitchell 90] Mitchell W. J., “The Logic of Architecture- Design, Computation, and Cognition”, MIT Press, 1990
- [Mitchell 98] Mitchell M., “An introduction to Genetic Algorithms”. MIT Press, Cambridge, Mass, 1998
- [Monedero 00] Monedero J., “Parametric Design: A Review and Some Experiences”, in Automation in Construction, vol. 9, no. 4, pp. 369-377. 2000
- [Moore 97] Moore, C. J., Miles, J.C., Rees, D. W. G., “Decision Support for Conceptual Bridge Design”, Artificial Intelligence in Engineering, v11, n3, Elsevier Science Ltd, Oxford, Engl., p.

- 259-272. 1997
- [Mühlenbein 92] Mühlenbein H., Darwin's continent cycle theory and its simulation by the Prisoner's Dilemma. *Complex Systems* 5, 459-478, 1992
- [Mühlenbein 97] Mühlenbein, H., "Genetic Algorithms", in *Local Search in Combinatorial Optimization*, Aarts, E., Lenstra, J. K. (eds), John Wiley & Sons, pp. 137-171, 1997
- [Ngo 02] Ngo D., Teo L. S., Byrne J. G., "Evaluating Interface Aesthetics" *Knowledge and Information Systems* 4: 46, 2002
- [O'Reilly 98] O'Reilly U-M., Ramachandran, G., "A preliminary investigation of evolution as a form design strategy", In Adami, C., Belew, R., Kitano, H. and Taylor C., (Eds), *Proceedings of the 6th conference on Artificial Life*, MIT press 1998
- [Osyczka 02] Osyczka A., "Evolutionary Algorithms for Single and Multicriteria Design Optimization", *Studies in Fuzziness and Soft Computing*, Physica-Verlag, Heidelberg, 218 pp. 2002
- [Pahl 96] Pahl, G. & Beitz, W., "Engineering Design: A Systematic Approach", (2 edition), Springer-Verlag, London, 1996
- [Park 98] Park K. W., Grierson D. E., "Pareto-optimal Conceptual Design of the Structural Layout of Buildings Using a Multicriteria Genetic Algorithm", *Journal of Computer-Aided Civil and Infrastructure Engineering*, Blackwell, 1998
- [Parmee 94] Parmee, I. C., (ed.) *Proceedings of the Adaptive Computing in Engineering Design and Control -'94*, PEDC, Plymouth, September 1994
- [Parmee 98] Parmee I. C., (eds.) "Adaptive Computer in Design and Manufacture", Engineering Design Centre, University of Plymouth, UK. Springer, 1998
- [Plemenos 02] Plemenos D., Miaoulis G., Vassilas N., "Machine learning for a General Purpose Declarative Scene Modeller". Intern. Conference GraphiCon 2002, Nizhny Novgorod, Russia 2002

- [Plemenos 91] Plemenos D., "Contribution to the study and development of techniques of modelling, generation and display of scenes. The MultiFormes project", Professorial dissertation, Nantes France, 1991 (in French).
- [Plemenos 93] Plemenos D., "Techniques for implementing learning mechanisms in a hierarchical declarative modeller". Research report MSI 93 - 04, Limoges France, (in French), 1993
- [Plemenos 95] Plemenos D., "Declarative Modelling by Hierarchical Decomposition. The actual state of the Multiform Project", Int. Conf. GraphiCon' 95, St Petersburg, Russia, 1995.
- [Plemenos 97] Plemenos D., Tamine K., "Increasing the efficiency of declarative modelling. Constraint evaluation for the hierarchical decomposition approach". International Conference WSCG'97, Plzen (Czech Republic), February 1997
- [Plemenos 98] Plemenos D., Ruchaud W., Tamine K., "Interactive techniques for declarative modelling". 3IA'98 International Conference, Limoges (France), 1998
- [Podehl 02] Podehl G., "Terms and Measures for Styling Properties", Proceedings of the 7th International Design Conference, May 14th - 17th 2002, Dubrovnik, Croatia, Zagreb. Sveucilisna tiskara 2002 pp. 879-886, 2002
- [Pohl 94] Pohl J., Myers L., "A Distributed Cooperative Model for Architectural Design", in Carrara G., Kalay Y. E., (eds.), Knowledge-Based Computer Aided Architectural Design, Elsevier Science Publishers, Amsterdam, The Netherlands, 1994
- [Pothorn 81] Pothorn H., 'Architectural Styles: An Historical Guide to World Design', Facts On File, New York. 1981
- [Poulet 96] Poulet F., Lucas M., "Modeling megalithic Sites", Proceeding of Eurographics'96, Poitiers, France, pp.279-288, 1996
- [Ravani 00] Ravani I., "Etude et développement de la notion de concept

- pour MultiCAD”, MSc Dissertation, Nantes, France 2000
- [Ravani 01] Ravani I., “The notion of concept in MultiCAD”, Rapport de recherche MSI 01-02 Université de Limoges Octobre 2001
- [Ravani 03] Ravani I., Makris D., Miaoulis G., Constantinides P., Petridis A., Plemenos D., “Implementation of Architecture-oriented Knowledge Framework in MultiCAD Declarative Scene Modelling System”, 1st Balcan Conference in Informatics, Thessaloniki, Greece, 2003
- [Ravani 04] Ravani J., Makris D., Miaoulis G., Plemenos D., Concept-Based Declarative Description Subsystem for Computer Aided Declarative Design (CADD). 3IA’04, Limoges, France 2004
- [Renner 03] Renner G., Ekart A., “Genetic algorithms in computer aided design” Computer-Aided Design 35 709–726, 2003
- [Rivard 00] Rivard H., Fenves S. J., “A Representation for Conceptual Design of Buildings”, Journal of Computing in Civil Engineering, 14 (3) 151-159, 2000
- [Rowe 91] Rowe P., “Design Thinking”, MIT Press, Cambridge, MA, USA, 1991
- [Ruchaud 01] Ruchaud W., “Etude et réalisation d’un moteur de résolution de contraintes géométriques pour la modélisation déclarative”, PhD Thesis, Limoges, 2001
- [Rush 86] Rush, R., The Building Systems Integration Handbook, American Institute of Architects, John Wiley & Sons, New York, 1986
- [Rychener 88] Rychener M. D., “Expert Systems for Engineering Design”, Academic Press, 1988
- [Sanchez 03] Sanchez S., le Roux O., Luga H., Gaildrat V., “Constraint-Based 3D-Object Layout using a Genetic Algorithm” 3IA’03 International Conference, Limoges, France 2003
- [Sequin 05] Sequin C., “CAD tools for aesthetic engineering” Computer-Aided Design 37 737–750, 2005
- [Shah 1995] Shah J., J., Mäntylä M., “Parametric and Feature-based

- CAD/CAM, Concepts, Techniques and Applications”, New York, John Wiley & Sons, 1995
- [Shapiro 61] Shapiro M, “Style”, in ‘Aesthetics Today’ Ed. M Philipson (Word Publishing, Cleveland, OH) pp 81-113. 1961
- [Shrestha 98] Shrestha, S. M., Ghaboussi, J., “Evolution of Optimum Structural Shapes Using Genetic Algorithm”, ASCE Journal of Structural Engineering, v124, n11, Reston, VA, USA, p. 1331-1338, 1998
- [Simon 75] Simon H. A., “Style in design”, in “Spatial Synthesis in Computer-aided Building Design” Ed. Eastman C., Applied Science, London, pp 287-309, 1975
- [Simon 96] Simon, H.A., “The Science of the Artificial”, MIT Press, Cambridge, MA, USA, 1996
- [Smith 95] Smith R., Warrington S., Mill F., “Shape representation for optimization”. Genetic Algorithms Engng Syst: Innovations Appl 112–7, 1995
- [Smithies 81] Smithies K. W., “Principles of Design in Architecture” Van Nostrand Reinhold, New York, 1981
- [Srinivas 95] Srinivas N., Deb K., “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”, Evolutionary Computation, v2:3, 221-248, 1995
- [Stahl 01] Stahl A., “Learning Feature Weights by Using Case Order Feedback”, Case-Based Reasoning Research and Development, Aha, D.W., Watson, I. & Yang, Q. (Eds.) Proc. of the 4th. International Conference on Case-Based Reasoning, ICCBR-2001, Springer, Lecture Notes in Artificial Intelligence, 2001
- [Staudek 02] Staudek T., “Exact Aesthetics. Object and Scene to Message” PhD. thesis. Masaryk University, Brno, Czechia 2002
- [Stiny 78] Stiny, G., Grips, J., “Algorithmic Aesthetics”, University of California Press Ltd. 1978
- [Stiny 78] Stiny G., Mitchell W.J., “The Palladian grammar” Environment and Planning B 5 pp 5-18, 1978

- [Sun 01] Sun J., "A Framework for Supporting Generative Product Design Using Genetic Algorithms" Ph D thesis Hong Kong Polytechnic University, Hong Kong, 2001
- [Syswerda 89] Syswerda, G., "Uniform Crossover in Genetic Algorithms". In Schaffer, D. (eds.), Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann Publication, 1989
- [Tappeta 99] Tappeta R. V., Renaud, J. E., "Interactive Multiobjective Optimization Procedure with Local Preferences", 3rd World Congress on Structural and Multidisciplinary Optimization (WCSMO-3), May 17-21 1999, Buffalo, New York 1999
- [Tsang 93] Tsang E., "Foundations of Constraint Satisfaction", Computation in Cognitive Science, Academic Press, 1993
- [Tzonis 94] Tzonis A., White I. (eds) "Automation Based Creative Design", Elsevier, Netherlands, pp.139-159, 1994
- [Tzonos 83] Tzonos P., "Typology of Habitat" (in Greek), Thessaloniki, Greece, 1983
- [van Veldhuizen 00] van Veldhuizen D. A., Lamont, G. B., "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art", Evolutionary Computation vol. 8 (2), pp. 125-147. 2000
- [van Veldhuizen 98] van Veldhuizen D. A., Lamont, G. B., "Multiobjective Evolutionary Algorithm Research : A History and Analysis", Air Force Institute of Technology, Wright-Patterson AFB, 1998
- [van Veldhuizen 99] van Veldhuizen D., A., Lamont G. B., "Multiobjective Evolutionary Algorithm Test Suites", Proceedings of the '99 ACM Symposium on Applied Computing, San Antonio, Texas, eds. J. Carroll, H. Haddad, D. Oppenheim, B. Bryant & G.B. Lamont, pp. 351-357, 1999
- [van Leeuwen 00] van Leeuwen J.P., de Vries B., "Modelling with Features and the formalisation of early design knowledge", Proceedings 3rd European Conference on Product and Process Modelling Lisbon, Portugal, September 25-27, 2000

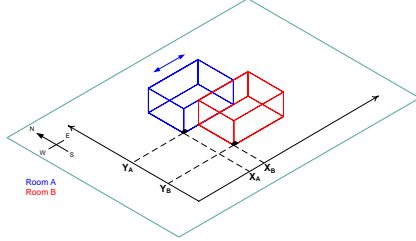
- [van Leeuwen 01] van Leeuwen J.P., Hendricx A., Fridqvist S., “Towards Dynamic Information Modelling in Architectural Design”, In: Proceedings CIB-W78 International Conference IT in Construction in Africa 2001, CSIR, Division of Building and Construction Technology, pp 19.1-14, 2001
- [van Leeuwen 97] van Leeuwen J.P., Wagter H., “Architectural design by-Features” In Proceedings CAAD Futures '97, p.97-115, Dordrecht, Kluwer, 1997
- [van Leeuwen 99] van Leeuwen J.P., “Modelling Architectural Design Information by Features”, PhD. Thesis, Eindhoven University of Technology, Eindhoven, 1999
- [de Ven 87] Cornelis van de Ven, “Space in Architecture – The evolution of a new idea in the theory and history of the modern movements”, Van Gorcum & Comp.B.B., Assen in the Netherlands, 1987
- [Vassilas 02] Vassilas V., Miaoulis G., Chronopoulos D., Konstantinidis E., Ravani I., Makris D., Plemenos D., “MultiCAD-GA: a system for the design of 3D forms based on genetic algorithms and human evaluation”, Lectures Notes on Artificial Intelligence (LNAI 2308), Springer-Verlag, p 203-214, 2002
- [Wang 02] Wang, C., Vergeest, J. S. M., et. al., “Cross model shape reuse: copying and pasting of freeform features”, Proceedings of DAC'02:2002 ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conferences, Montreal, Canada, September 29, 2002 - October 2, 2002
- [Wang 03] Wang, C., Vergeest, J.S.M., Wiegers, T. D. J., van der Pant T. M. C. van den Berg, , “Exploring the Influence of Feature Geometry on Design Style”, Electrical and Computer Engineering Series: «Computational Methods in Circuits and Systems Application», Mastorakis, N.E., et al. (ed.), pp.21-28, WSEAS Press, 2003

- [Woodbury 95] Woodbury, R., Chang, T.-W., “Massing and enclosure design with SEED-Config”, *Journal of Architectural Engineering*, ASCE 1 (4) 170-178, 1995
- [Zitzler 00] Zitzler E., Thiele L., Deb K., “Comparison of Multiobjective Evolutionary Algorithms Empirical Results”, *Evolutionary Computation*, vol. 8 (2), pp. 173-195, 2000
- [Zitzler 99] Zitzler E., Thiele L., “Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach”, *IEEE Transactions on Evolutionary Computation*, vol. 3 (4), pp. 257-271 1999

Appendix

Definition of Constraints

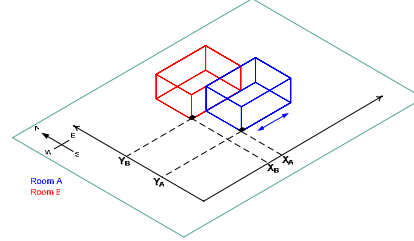
Room A Adjacent North of Room B



$$X_B - L_A - D_{AN} \leq X_A \leq X_B + L_B - D_{AN}$$

$$Y_B = Y_A + W_A$$

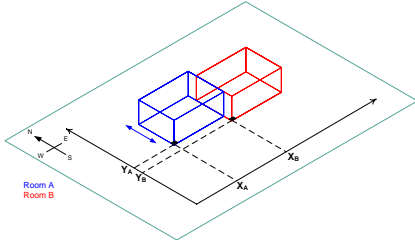
Room A Adjacent South of Room B



$$X_B - L_A - D_{AN} \leq X_A \leq X_B + L_B - D_{AS}$$

$$Y_A = Y_B + W_A$$

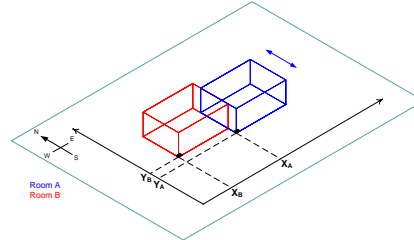
Room A Adjacent West of Room B



$$Y_B - W_A - D_{AW} \leq Y_A \leq Y_B + W_B - D_{AW}$$

$$X_A = X_B - L_A$$

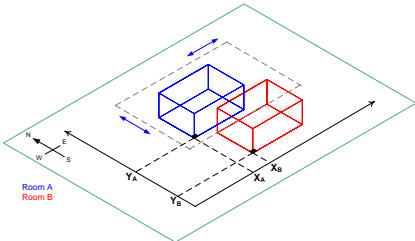
Room A Adjacent East of Room B



$$Y_B - W_A - D_{AE} \leq Y_A \leq Y_B + W_B - D_{AE}$$

$$X_A = X_B + L_B$$

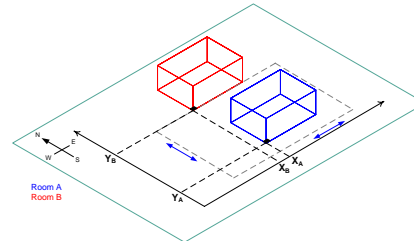
Room A Near North of Room B



$$(Y_B + W_B) - W_B \leq Y_A \leq (Y_B + W_B) + \frac{2}{3}W_A$$

$$(X_B - \frac{2}{3}L_A) \leq X_A \leq (X_B + L_B) - \frac{2}{3}L_A$$

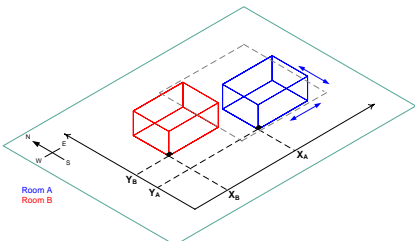
Room A Near South of Room B



$$(Y_B - W_A) - \frac{2}{3}W_A \leq Y_A \leq (Y_B - W_A)$$

$$(X_B - \frac{2}{3}L_A) \leq X_A \leq (X_B + L_B) - \frac{2}{3}L_A$$

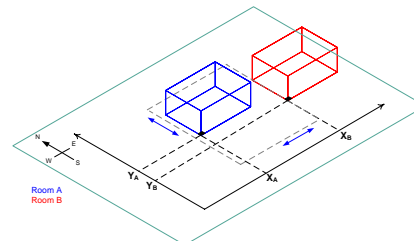
Room A Near East of Room B



$$(Y_B - W_B) \leq Y_A \leq (Y_B + W_B) - \frac{2}{3}W_A$$

$$(X_B + L_B) \leq X_A \leq (X_B + L_B) + \frac{2}{3}L_A$$

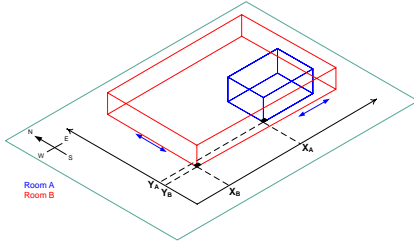
Room A Near West of Room B



$$(Y_B - W_A) \leq Y_A \leq (Y_B + W_B) + \frac{2}{3}W_A$$

$$(X_B - L_A) - \frac{2}{3}L_A \leq X_A \leq (X_B - L_A)$$

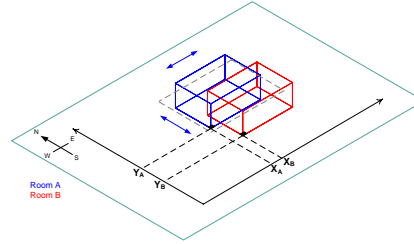
Room A Inside of Room B



$$Y_B \leq Y_A \leq (Y_B + W_A) - W_A$$

$$X_B \leq X_A \leq (X_B + L_B) - L_A$$

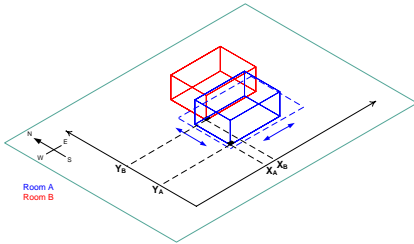
Room A Overlap North of Room B



$$(Y_B - \frac{1}{2}W_A) - L_B \leq Y_A \leq (Y_B + W_A) - d_{ON}$$

$$(X_B - L_A) + d_{ON} \leq X_A \leq (X_B + L_B) - d_{ON}$$

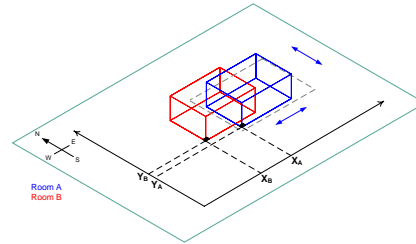
Room A Overlap South of Room B



$$(Y_B - W_A) + d_{OS} \leq Y_A \leq Y_B + \frac{1}{2}W_B$$

$$(X_B - L_A) + d_{OS} \leq X_A \leq (X_B + L_B) - d_{OS}$$

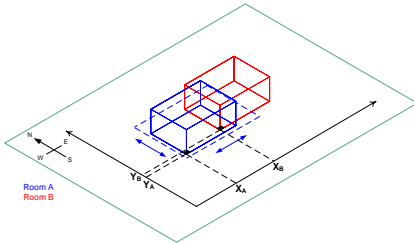
Room A Overlap East of Room B



$$(Y_B - W_A) + d_{OE} \leq Y_A \leq (Y_B + W_B) + d_{OE}$$

$$(X_B - \frac{1}{2}L_B) + d_{OE} \leq X_A \leq (X_B + L_B) - d_{OE}$$

Room A Overlap West of Room B

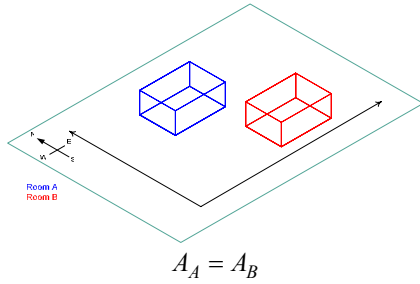


$$(Y_B - W_B) + d_{OW} \leq Y_A \leq (Y_B + W_B) - d_{OW}$$

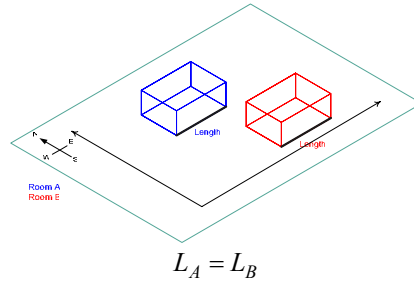
$$(X_B - L_A) + d_{OW} \leq X_A \leq (X_B + \frac{1}{2}L_B)$$

Definition of Fitness

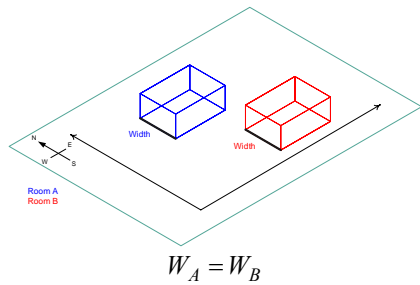
Room A Same Area Room B



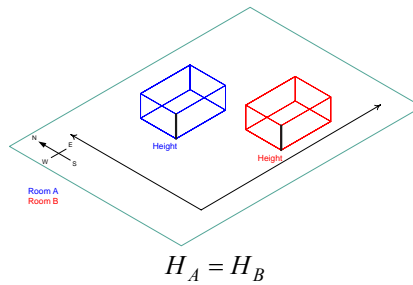
Room A Same Length Room B



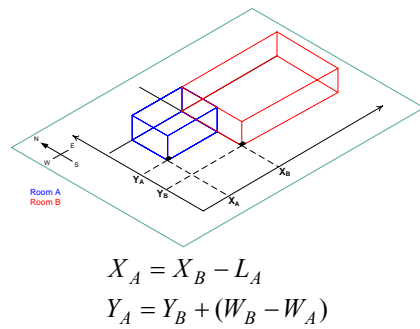
Room A Same Width Room B



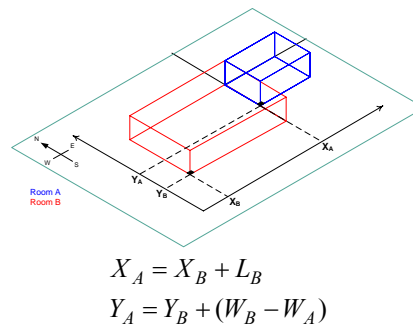
Room A Same Height Room B



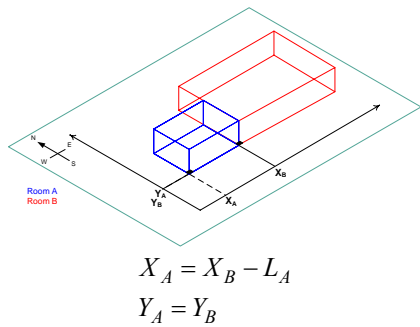
Room A Align West North of Room B



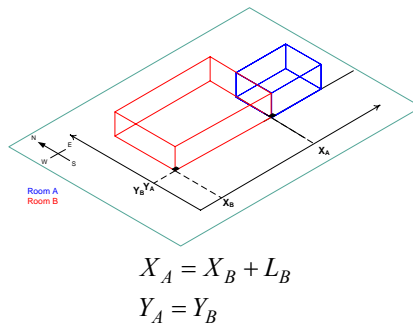
Room A Align East North of Room B



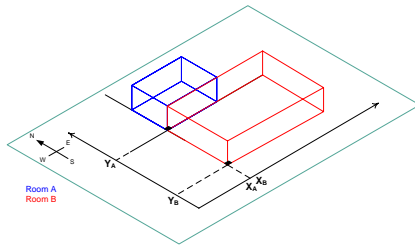
Room A Align West South of Room B



Room A Align East South of Room B



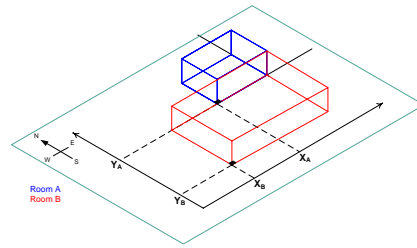
Room A Align North West of Room B



$$X_A = X_B$$

$$Y_A = Y_B + W_B$$

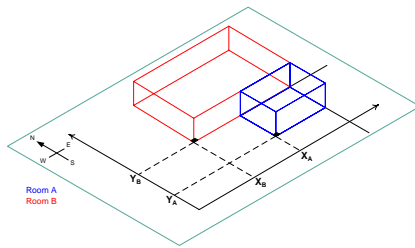
Room A Align North East of Room B



$$X_A = X_B + (L_B - L_A)$$

$$Y_A = Y_B + W_B$$

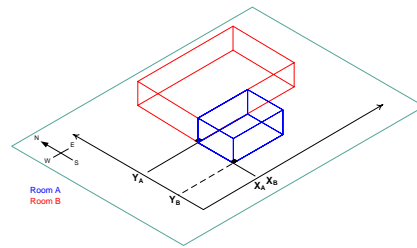
Room A Align South East of Room B



$$X_A = X_B + (L_B - L_A)$$

$$Y_A = Y_B - W_A$$

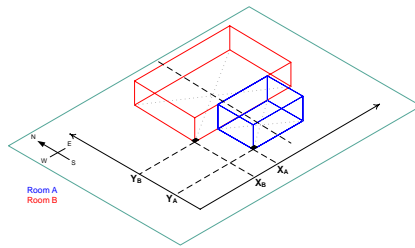
Room A Align South West of Room B



$$X_A = X_B$$

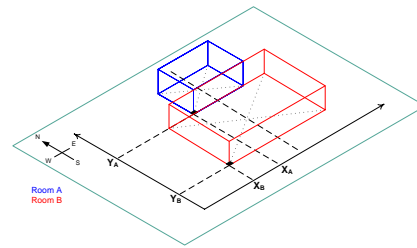
$$Y_A = Y_B - W_A$$

Room A Axial Centre South North of Room B Room A Axial Centre North South of Room B



$$(X_B - \frac{1}{2}L_B) - (X_A - \frac{1}{2}L_A) = 0$$

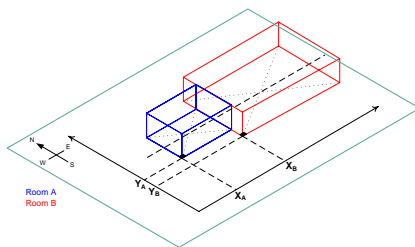
$$Y_B \geq Y_A + W_A$$



$$(X_B + \frac{1}{2}L_B) - (X_A + \frac{1}{2}L_A) = 0$$

$$Y_B \geq Y_A - W_A$$

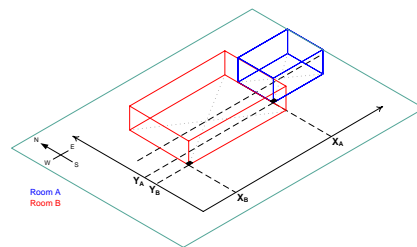
Room A Axial East West of Room B



$$(Y_B + \frac{1}{2}W_B) - (Y_A + \frac{1}{2}W_A) = 0$$

$$X_B \leq X_A - L_B$$

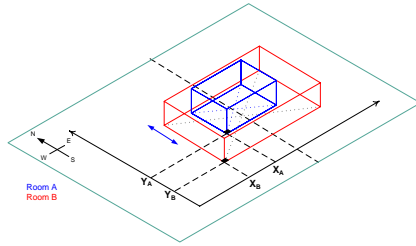
Room A Axial West East of Room B



$$(Y_B + \frac{1}{2}W_B) - (Y_A + \frac{1}{2}W_A) = 0$$

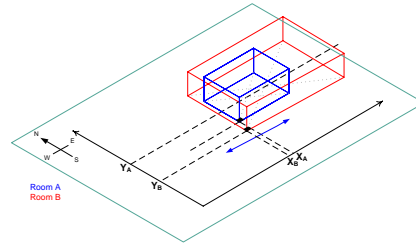
$$X_B \geq X_A + L_A$$

Room A Axial Centre Vertical Room B



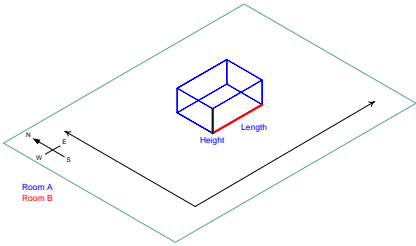
$$(X_B + \frac{1}{2}L_B) - (X_A + \frac{1}{2}L_A) = 0$$

Room A Axial Centre Horizontal Room B



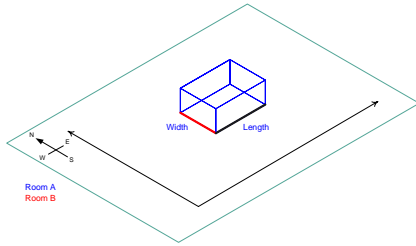
$$(Y_B + \frac{1}{2}W_B) - (Y_A + \frac{1}{2}W_A) = 0$$

Room A Higher than Long



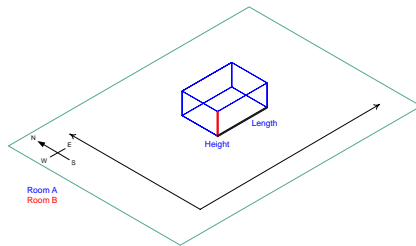
$$H_A > L_A$$

Room A Longer than Deep



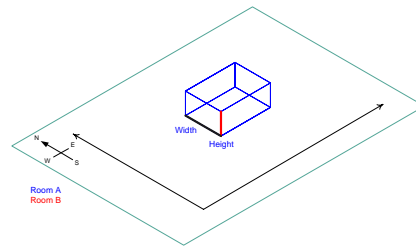
$$L_A > D_A$$

Room A Longer than High



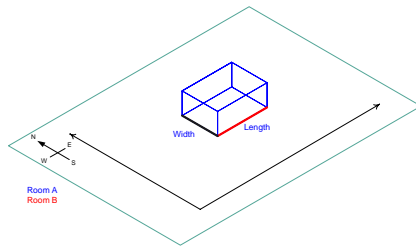
$$L_A > H_A$$

Room A Deeper than High



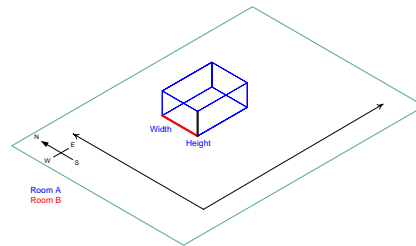
$$D_A > H_A$$

Room A Deeper than Long



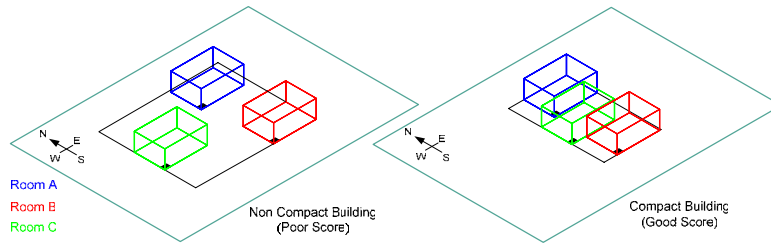
$$D_A > L_A$$

Room A Higher than Deep

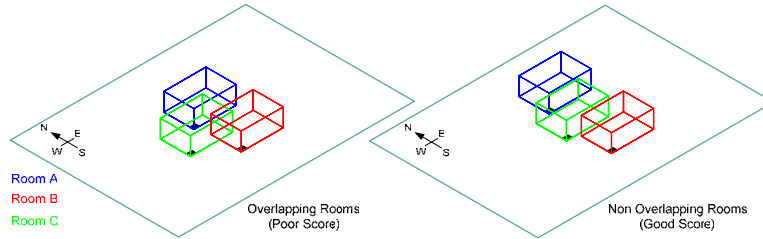


$$H_A > D_A$$

Building Compactness

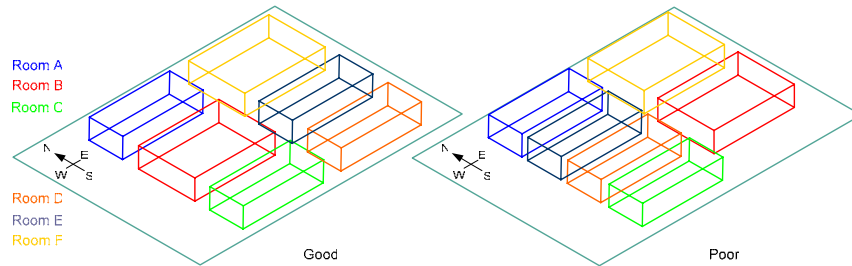


Non Overlapping



Aesthetic evaluation

Balance



$$BQ = 1 - \frac{|BQ_{vert}| + |BQ_{hor}|}{2} \in [0, 1]$$

BQ_{vert} and BQ_{hor} are, respectively, the vertical and horizontal balances.

$$BQ_{vert} = \frac{WL - WR}{\max(|WL|, |WR|)}$$

$$BQ_{hor} = \frac{WT - WB}{\max(|WT|, |WB|)}$$

$$w_j = \sum_i^{n_j} a_{ij} d_{ij} \quad j = L, R, T, B$$

Where :

L : left quadrant of the bounding box;

R : right quadrant of the bounding box;

T : top quadrant of the bounding box;

B : bottom quadrant of the bounding box;

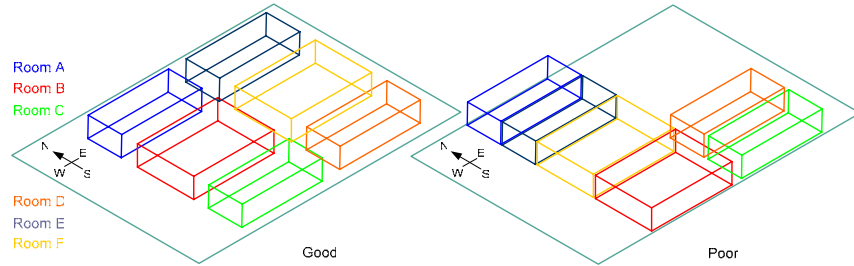
w_j : total weight of side j ;

a_{ij} : area of object i on side j ;

d_{ij} : distance between the central lines of the object and the bounding box;

n_j : total number of objects on the side.

Equilibrium



$$EQ = 1 - \frac{|EQx| + |EQy|}{2}, [0, 1]$$

EQx : normalized x-coordinate of the center of mass of the objects.

$$EQx = \frac{2 \sum_i^n a_i (x_i - x_c)}{l_{bb} \sum_i^n a_i}$$

EQy : normalized y-coordinate of the center of mass of the objects.

$$EQy = \frac{2 \sum_i^n a_i (y_i - y_c)}{w_{bb} \sum_i^n a_i}$$

Where:

(x_i, y_i) : coordinates of the centers of object i ;

(x_c, y_c) : coordinates of the center of the bounding box;

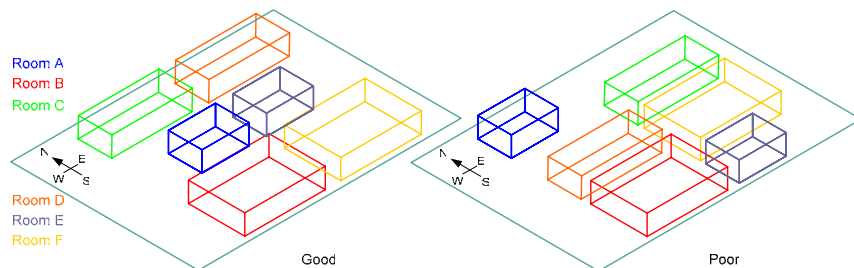
a_i : area of the object;

l_{bb} : length of the bounding box;

w_{bb} : width of the bounding box;

n : the number of objects within the bounding box.

Symmetry



$$SYQ = 1 - \frac{|SYQ_{vert}| + |SYQ_{hor}| + |SQ_{rad}|}{3} \in [0, 1]$$

SYQ_{vert} , SYQ_{hor} , and SYQ_{rad} are, respectively, the vertical, horizontal, and radial symmetries.

$$SYQ_{vert} = \frac{|X'_{UL} - X'_{UR}| + |X'_{LL} - X'_{LR}| + |Y'_{UL} - Y'_{UR}| + |Y'_{LL} - Y'_{LR}| + |H'_{UL} - H'_{UR}| + |H'_{LL} - H'_{LR}| + |B'_{UL} - B'_{UR}| + |B'_{LL} - B'_{LR}| + |\Theta'_{LL} - \Theta'_{UR}| + |\Theta'_{LL} - \Theta'_{LR}| + |R'_{UL} - R'_{UR}| + |R'_{LL} - R'_{LR}|}{12}$$

$$SYQ_{hor} = \frac{|X'_{UL} - X'_{LL}| + |X'_{UR} - X'_{LR}| + |Y'_{UL} - Y'_{LL}| + |Y'_{UR} - Y'_{LR}| + |H'_{UL} - H'_{LL}| + |H'_{UR} - H'_{LR}| + |B'_{UL} - B'_{LL}| + |B'_{UR} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{LL}| + |\Theta'_{UR} - \Theta'_{LR}| + |R'_{UL} - R'_{LL}| + |R'_{UR} - R'_{LR}|}{12}$$

$$SYQ_{rad} = \frac{|X'_{UL} - X'_{LR}| + |X'_{UR} - X'_{LL}| + |Y'_{UL} - Y'_{LR}| + |Y'_{UR} - Y'_{LL}| + |H'_{UL} - H'_{LR}| + |H'_{UR} - H'_{LL}| + |B'_{UL} - B'_{LR}| + |B'_{UR} - B'_{LL}| + |\Theta'_{LL} - \Theta'_{LR}| + |\Theta'_{UR} - \Theta'_{LL}| + |R'_{UL} - R'_{LR}| + |R'_{UR} - R'_{LL}|}{12}$$

X'_j , Y'_j , H'_j , B'_j , Θ'_j , and R'_j are, respectively, the normalized values of

$$X_j = \sum_i^{n_j} |x_{ij} - x_c| \quad j = UL, UR, LL, LR$$

$$Y_j = \sum_i^{n_j} |y_{ij} - y_c| \quad j = UL, UR, LL, LR$$

$$H_j = \sum_i^{n_j} h_{ij} \quad j = UL, UR, LL, LR$$

$$B_j = \sum_i^{n_j} b_{ij} \quad j = UL, UR, LL, LR$$

$$\Theta_j = \sum_i^{n_j} \left| \frac{y_{ij} - y_c}{x_{ij} - x_c} \right| \quad j = UL, UR, LL, LR$$

$$R_j = \sum \sqrt{(x_{ij} - x_c)^2 + (y_{ij} - y_c)^2} \quad j = UL, UR, LL, LR$$

Where :

UL: upper-left;

UR: upper-right;

LL: lower-left;

LR: lower-right;

X_j : total x-distance of quadrant j;

Y_j : total y-distance;

H_j : total height;

B_j : total width;

Θ_j : total angle;

R_j : total distance;

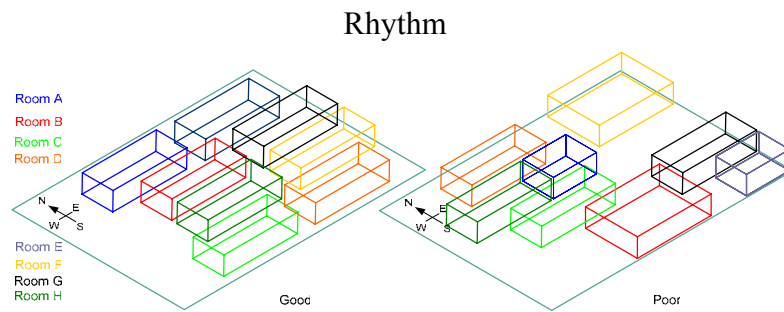
(x_{ij}, y_{ij}) : coordinates of the centers of object i on quadrant j ;

(x_c, y_c) : coordinates of the centers of the frame;

b_{ij} : width of the object;

h_{ij} : height of the object;

n_j : total number of objects on the quadrant.



$$RTHQ = 1 - \frac{|RTHQ_x| + |RTHQ_y| + |RTHQ_{area}|}{3}$$

$$RTHQ_x = \frac{|X'_{UL} - X'_{UR}| + |X'_{UL} - X'_{LR}| + |X'_{UL} - X'_{LL}| + |X'_{UR} - X'_{LR}| + |X'_{UR} - X'_{LL}| + |X'_{LR} - X'_{LL}|}{6}$$

$$RTHQ_y = \frac{|Y'_{UL} - Y'_{UR}| + |Y'_{UL} - Y'_{LR}| + |Y'_{UL} - Y'_{LL}| + |Y'_{UR} - Y'_{LR}| + |Y'_{UR} - Y'_{LL}| + |Y'_{LR} - Y'_{LL}|}{6}$$

$$RTHQ_{area} = \frac{|A'_{UL} - A'_{UR}| + |A'_{UL} - A'_{LR}| + |A'_{UL} - A'_{LL}| + |A'_{UR} - A'_{LR}| + |A'_{UR} - A'_{LL}| + |A'_{LR} - A'_{LL}|}{6}$$

X'_j , Y'_j , and A'_j are, respectively, the normalized values of

$$X_j = \sum_i^{n_j} |x_{ij} - x_c| \quad j = UL, UR, LL, LR$$

$$Y_j = \sum_i^{n_j} |y_{ij} - y_c| \quad j = UL, UR, LL, LR$$

$$A_j = \sum_i^{n_j} a_{ij} \quad j = UL, UR, LL, LR$$

Where:

UL: upper left quadrant;

UR: upper right quadrant;

LL: lower left quadrant;

LR: lower right quadrant;

X_j : total x -distance of quadrant j ;

Y_j : total y -distance of quadrant j ;

A_j : is the total area;

(x_{ij}, y_{ij}) : coordinates of the centers of object i on quadrant j ;

(x_c, y_c) : coordinates of the centers of the frame;

a_{ij} : area of the object;

n_j : total number of objects on the quadrant.

