UNIVERSITÉ DE LIMOGES

École Doctorale Science – Technologie – Santé

Faculté des Sciences & Techniques

Laboratoire XLIM

Thèse N°

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Informatique

Présentée et soutenue publiquement par

Georgios P. Bardis

le 26 Juin 2006

APPRENTISSAGE ET AIDE À LA DÉCISION POUR LA MODÉLISATION DÉCLARATIVE DE SCÈNES

Acquisition et gestion des préférences morphologiques dans le cadre d'un environnement de conception

Thèse dirigée par Professeur Dimitri PLEMENOS

Co-encadrement Professeur Georges MIAOULIS

Jury

Président :

Djamchid GHAZANFARPOUR Professeur Université de Limoges XLIM

Rapporteurs :

Véronique GAILDRAT Maître de conférences HDR Université Paul Sabatier Toulouse III

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

Examinateurs :

Georges MIAOULIS Professeur Institut d'Education Technologique d'Athènes

Dimitri PLEMENOS Professeur Université de Limoges XLIM

Je dédie ce travail à tous qui,
pendant ces années,
m'ont soutenu par leur amour.

Remerciements

Je souhaite remercier Madame Véronique Gaildrat, Maître de Conférences HDR à l'Université Paul Sabatier de Toulouse, et Monsieur Nikolaos Vassilas, Professeur Associé d'Informatique à l'Institut d'Education Technologique d'Athènes, pour avoir bien voulu rapporter sur cette thèse, et pour leurs remarques pertinentes qui ont permis d'améliorer ce travail.

Je remercie Monsieur Djamchid Ghazanfarpour, Professeur d'Informatique à l'Université de Limoges, de bien vouloir présider ce jury, et de me faire l'honneur d'être présent à la soutenance.

Je remercie aussi Monsieur Georges Miaoulis, Professeur d'Informatique à l'Institut d'Education Technologique d'Athènes, et Monsieur Dimitri Plemenos, Professeur d'Informatique à l'Université de Limoges, pour avoir fait partie du jury lors de la soutenance de cette thèse.

Je tiens à exprimer ma reconnaissance tout particulièrement à Professeurs Georges Miaoulis et Dimitri Plemenos pour leurs conseils concernant mon travail de recherche ainsi que pour leur soutien patient et précieux pendant ces années. Je les remercie pour m'avoir donné la possibilité de réaliser ce travail.

Je remercie aussi Jean Dragonas, Vassilis Golfinopoulos, Dimitris Makris et Ioanna Ravani pour leur collaboration et leurs encouragements.

Je remercie également Yannis Havoutis et Vassilis Stathopoulos pour leur collaboration et contribution à l'implémentation de ce travail de recherche.

Résumé

La méthodologie de *Modélisation Déclarative* offre au concepteur la possibilité de décrire un objet ou un environnement en utilisant des termes abstraits, au lieu des valeurs explicites pour des propriétés géométriques concrètes. L'avantage principal de cette approche, particulièrement pendant la première phase de conception, est de permettre au concepteur de se concentrer sur des caractéristiques qui sont plus près à l'intuition humaine. La Modélisation Déclarative peut proposer des solutions acceptables, souvent originales, aux quelles le concepteur n'aurait peut-être pas pensé. D'autre part, son désavantage principal provient du fait qu'une description déclarative mène généralement à un grand nombre de représentations géométriques alternatives, toutes *solutions* légitimes, dans le sens de la conformité avec la description soumise, pourtant pas équitablement préférées par le concepteur.

Le but du présent travail a été l'étude et l'implémentation d'une méthodologie pour l'acquisition et la représentation des préférences du concepteur sous une forme de modèle informatique. Un tel modèle devrait être capable de fournir un environnement déclaratif de conception avec un comportement intelligent en ce qui concerne *les préférences d'utilisateur*. La méthodologie proposée, combine des éléments de domaine *de l'apprentissage automatique* et *de l'aide à la décision* offrant deux modèles respectifs de préférence. Le noyau du premier modèle est un mécanisme d'apprentissage incrémental, basé sur l'évaluation des solutions par l'utilisateur, durant l'utilisation régulière du système, tandis que le deuxième comporte un ensemble des vecteurs, représentant l'importance et la graduation d'une série d'attributs observés selon l'utilisateur spécifique.

Un Module Intelligent de Profil d'Utilisateur, qui implémente les deux modèles de préférence et les composants additionnels exigés par le système d'information, a été intégré à un environnement déclaratif de conception, permettant la comparaison et l'essai. Une série d'expériences a été entreprise, avec différentes configurations du module et différents profils d'utilisateur, donnant des résultats qui suggèrent qu'un environnement déclaratif de conception puisse tirer bénéfice de l'application des modèles de préférence proposés.

Abstract

The *Declarative Modelling* methodology offers the designer the ability to describe an object or an environment using abstract terms instead of explicit values for concrete geometric properties. The major advantage of this approach, especially during early-phase design, is that it allows concentration on characteristics which are closer to human intuition. Moreover, it may yield acceptable geometric representations not originally conceived by the designer. On the other hand, its main disadvantage stems from the fact that a moderate declarative description generally leads to a large number of alternative geometric representations, all legitimate *solutions,* in the sense of compliance with the submitted description, yet not equally preferable on behalf of the designer.

The aim of the current work has been the study and implementation of a methodology for the acquisition and representation of the designer's preferences in the form of a computational model. Such a model should be capable of providing a declarative design environment with intelligent behaviour features with respect to *user preferences*. The herein proposed and implemented methodology combines elements from the areas of *machine learning* and *decision support* offering two respective preference models. The core of the former is an incrementally learning mechanism, based on user solution evaluation during regular system use, whereas the latter comprises a set of vectors, representing the importance and scaling for a series of observed attributes according to the specific user.

An Intelligent User Profile Module, implementing both preference models and the required additional information system components, has been integrated to a declarative design environment, allowing comparison and testing. A series of experiments has been conducted, with respect to alternative module settings and user profiles yielding results suggesting that a declarative design environment may benefit from the application of the proposed preference models.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Introduction

Introduction

Nous allons tracer ici les grandes lignes des considérations qui ont motivé ce travail, définir ses objectifs et discuter brièvement du contexte dans lequel il a été réalisé ainsi des grands domaines scientifiques qui ont contribué à sa réalisation.

Nous commençons par la suggestion que le problème que nous abordons ici est un cas spécial d'un problème déjà existant dans notre vie quotidienne et qui provient des performances constamment et rapidement améliorées des ordinateurs et des communications de données modernes: celui du débordement de l'information. Ensuite, nous procédons à une brève présentation de la méthodologie de la *Modélisation Déclarative*, des besoins qu'elle aborde et des problèmes inhérents qu'elle expose. Nous énonçons notre but et présentons un résumé de la solution que nous avons proposée et qui est mise en application dans le cadre de ce travail. Dans les sections restantes nous fournissons une vue d'ensemble des méthodologies et des domaines de recherche qui ont contribué à cette solution.

Habiter dans la bibliothèque

Une grande partie de notre vie quotidienne bénéficie, directement ou indirectement, de l'utilisation de l'ordinateur et des communications modernes de données. Les écoles et les universités utilisent cette combinaison pour des tâches fondamentales comme le traitement des textes et les présentations ainsi que pour la recherche avancée et pour l'enseignement à distance. Les entreprises effectuent également des tâches de recherche et de développement aussi bien que des tâches de prise de décision avec leur aide, tout en maintenant de vastes quantités de données concernant le marché, les finances et les clients et en effectuent du management à distance. Les transports bénéficient de l'aide des ordinateurs et des communications pour la coordination et le contrôle, l'art a trouvé en eux de nouveaux moyens d'expression, la médecine est largement aidée dans des tâches avancées de diagnostic, de thérapie et de conception de médicaments. Les ordinateurs et les communications se sont installés pratiquement dans tous les aspects de l'activité humaine.

Naturellement, il y a plus que des raisons adéquates pour cette mainmise. Les ordinateurs, d'une part, portent des qualités spécifiques jusqu'à un degré bien au delà de la capacité humaine. Ce sont ces qualités qui en font réellement un des outils les plus souples et les plus attrayants jamais conçus. Si nous essayions d'isoler les propriétés de base sur lesquelles ces qualités sont construites, nous observerions que les ordinateurs offrent, essentiellement, de la *vitesse* et de la *fiabilité* exceptionnelles concernant le traitement et l'enregistrement de données. Les communications modernes de données, d'autre part, offrent la capacité de transférer d'une façon rapide et efficace, souvent en temps réel, de grands volumes d'information sous forme d'images, de sons, de vidéo et de documents, mettant en valeur de ce fait les capacités des ordinateurs, les rendant pratiquement omniprésents dans notre vie quotidienne.

Les innovations technologiques, pendant les vingt dernières années, ont sérieusement influencé le progrès de l'industrie des ordinateurs et des télécommunications, améliorant de façon spectaculaire les qualités mentionnées ci-dessus. En ce qui concerne les ordinateurs, le matériel dont l'utilisateur moyen pourrait seulement rêver pendant les années '80 peut maintenant être trouvé sur chaque bureau. Une configuration d'ordinateur personnel typique du milieu des années '80 comprenait une UC fonctionnant à 4,77 mégahertz, effectuant approximativement 1 million d'opérations en virgule flottante par seconde (1 MFLOP), 640 Ko de Mémoire Vive et un lecteur de disquettes souples pour des disques 5,25" de capacité de 360 Ko, accompagnés d'un moniteur monochrome de tube cathodique. En même temps, les communications pauvres de données pouvaient seulement offrir la voix et un transfert lent de texte à l'utilisateur moyen. Ces dispositifs, bien qu'objectivement non négligeables, n'étaient dans la pratique pas adéquats pour des applications de grande envergure. Aujourd'hui, une station de travail de même prix comporte une UC fonctionnant à 3,4 gigahertz, à 1 gigaoctet de Mémoire Vive et à un port USB 2.0 supportant les cartes de mémoire flash de capacité de 256 Mo ou plus, typiquement accompagnés d'un écran couleur TFT capable d'afficher quelques milliards de couleurs. Par ailleurs, les taux de transfert de 512Kbps et des interfaces utilisateur de haut niveau rendent le transfert de données, de n'importe quelle forme, rapide et facile. Ces nombres impliquent que les machines contemporaines représentent une amélioration d'environ 1000 fois sur chaque aspect de performance – voir par exemple les fréquences de l'UC mentionnées ci-dessus – et, bien plus impressionnant, elles sont comparables aux ordinateurs géants d'une valeur de millions des dollars des années '80 tel le célèbre Cray II. Grâce à ces avancées, la population d'utilisateurs d'ordinateur a rapidement augmenté. Les vieux besoins ont été transformés à de nouvelles demandes d'utilisateurs qui doivent être accomplies par ces machines puissantes et des canaux de télécommunication. Néanmoins, comme c'est presque toujours le cas, il faut plus que la puissance de calcul brute et la vitesse de transfert pour accomplir ces besoins d'une façon efficace.

J.L. Borges, dans sa courte histoire excellente appelée *Bibliothèque de Babel*, présente une métaphore qui a beaucoup de similitudes avec l'ère moderne de l'information. L'histoire fait partie d'une collection appelée *Labyrinthes* [Borges92] et a été pour la première fois éditée dans les années '40, mais elle semble être remarquablement prédictive du futur. L'endroit est l'intérieur d'un bâtiment énorme d'une architecture compliquée qui ressemble à un labyrinthe. Les salles de ce bâtiment sont reliées par des couloirs étroits ou des escaliers et les murs de chaque pièce sont couverts d'étagères remplies de livres. L'histoire, en fait, tourne autour de ces livres: chaque livre contient une séquence aléatoire de lettres couvrant sa première jusqu'à sa dernière page et, comme c'est normal, la plupart du temps le livre entier ne semble avoir aucun sens. Les gens, qui vivent et agissent dans la bibliothèque durant toute leur vie, cherchent désespérément un certain livre unique, supposé contenir l'explication de tous autres livres ainsi que de l'existence de la bibliothèque elle-même. Parfois, pendant leur recherche, ils trébuchent sur un mot ou une expression signicatifs qui apparaissent parmi les pages incompréhensibles d'un livre. En conséquence, ils développent différentes manières de penser basées sur ces petits fragments de raison aléatoirement placés. Un aspect impressionnant de l'histoire c'est l'implication qu'une partie des livres les plus importants qui ont jamais été écrits ou qui seront écrits peut également être trouvé dans la bibliothèque simplement parce que, après tout, ce sont juste des séquences de lettres. Cependant, à cause de la fraction minuscule qu'ils représentent contre tous les ordres possibles de lettres, c'est extrêmement difficile de les découvrir. Par ailleurs, nous restons avec le sentiment que des gens qui ont vécu parmi des pages aléatoires et incompréhensibles pendant des années, puissent ne pas apprécier et tirer profit de ces livres vraiment importants même s'ils les découvraient.

L'analogie avec le débordement de l'information d'aujourd'hui, bien que non intentionnelle, est étonnante. A cause des progrès technologiques mentionnés plus haut, nous sommes entourés de vastes quantités d'information, disponible sur le bout de nos doigts ou à quelques pas plus loin sous forme de sites Web, de livres, de journaux, de magazines, d'émissions de TV, de titres multimédia ou de publications. Néanmoins, nous devons exercer notre jugement presque constamment, contre des medias et des approches divers, afin de pouvoir filtrer le flot d'information entrant et le transformer en un sous-ensemble acceptable de connaissance ou de divertissement non seulement en termes de taille mais également en termes de qualité. La plupart du temps, cette sélection n'est pas – et ne peut pas être – objective: habituellement nous *filtrons* la partie de l'information entrante qui satisfait à nos besoins à l'instant donné, ignorant les parties qui pourraient être intéressantes pour d'autres. En tout cas, au lieu de choisir le meilleur, nous choisissons simplement ce que nous considérons comme le meilleur *pour nous,* tendant à filtrer cette partie de l'entrée qui est plus près de nos *préférences*.

Modélisation Déclarative et première phase de conception

La Modélisation Déclarative est une méthodologie puissante visant à soulager le créateur de la tâche pénible d'énoncer explicitement les propriétés géométriques d'un objet (ou d'un environnement) pendant la première phase de conception. À ce stade, le créateur s'intéresse plus aux idées et aux conditions fonctionnelles, recherchant l'intuition associée à l'utilisation pratique. Par conséquent, la Modélisation Déclarative offre au créateur la possibilité de décrire un objet à travers un ensemble de propriétés et de relations ambiguës qui pourraient être interprétées de plusieurs façons en ce qui concerne les résultats géométriques finals. Cette flexibilité permet au créateur d'examiner ces objets alternatifs, tous conformes à la description abstraite, et de choisir peut-être par la suite celui ou ceux étant plus proches de son intuition.

Cependant, il y a un prix à payer puisque l'avantage de la flexibilité et de la liberté pendant la conception d'un objet est traduit en un désavantage en termes de calcul pendant la génération et la visualisation des solutions. En fait, tous les objets conformes à la description suggérée, ne sont pas intéressants à niveau égal. Les versions alternatives peuvent être très nombreuses et le créateur n'est pas toujours capable ou prêt à les examiner toutes afin de sélectionner les versions les plus intéressantes. On a proposé plusieurs approches pour aborder ce problème basées sur les réseaux neuronaux, les algorithmes génétiques, le dialogue supplémentaire avec l'utilisateur, les exemples de représentations géométriques ou les combinaisons de ces mécanismes.

L'objectif

L'objectif principal de ce travail est de *fournir un mécanisme capable de choisir et de présenter à l'utilisateur le sous-ensemble des solutions, correspondant à une description déclarative donnée, le plus proche des préférences de l'utilisateur spécifique.*

La méthodologie proposée

Afin d'accomplir l'objectif mentionné ci-dessus, nous avons proposé et implémenté un module intelligent de profil d'utilisateur dans le contexte de l'environnement de conception déclarative Open-MultiCAD. Ce module est responsable de l'acquisition et du maintien des préférences de l'utilisateur dans une base de données dédiée en appliquant deux mécanismes alternatifs:

- Un mécanisme d'*apprentissage automatique* transparent, entraîné par des sélections réelles de l'utilisateur pendant l'utilisation régulière du système.
- Un mécanisme *d'aide à la décision*, initialisé explicitement par l'utilisateur pour refléter ses préférences personnelles.

Ces préférences sont invoquées chaque fois qu'une description déclarative est donnée par l'utilisateur et traitée par le système, qui présente ainsi seulement le sous-ensemble le plus intéressant, c'est-à-dire le sous-ensemble préféré de la population de solutions générée.

Domaines de recherches connexes

Ce travail et le prototype correspondant réalisé, utilisent des notions d'un éventail de domaines connexes. Une représentation visuelle du chevauchement de ces domaines apparaît sur la Figure 1.

L'Analyse de Décision est un exemple d'un domaine où les préférences et l'expertise du décideur doivent être traduites en un modèle significatif qui permettra la sélection des options les appropriées ou préférables parmi un certain nombre d'options alternatives. Plusieurs mécanismes ont été développés pour faciliter ce transfert de l'intuition à la représentation concrète correspondante. L'échange que les chercheurs doivent habituellement résoudre dans cette région est celui entre la rétroaction adéquate de l'utilisateur et sa surcharge minimale en termes de temps et de complexité concernant l'acquisition de cette rétroaction. Par exemple, il existe des méthodes qui exigent des comparaisons par paires de toutes les alternatives en ce qui concerne chaque attribut influant sur la sélection globale. Les comparaisons par paires constituent une des méthodes des plus largement acceptées et des plus directes pour obtenir les préférences de l'utilisateur, puisque leur adoption contribue à la simplicité. D'autre part, quoique simples et directes pour l'utilisateur, ces méthodes deviennent impraticables quand plus de quelques dizaines d'options alternatives doivent être évaluées puisque le nombre de paires à comparer devient prohibitif en termes de temps.

Quelques méthodes réduisent ce besoin de comparaison des options alternatives par paires en le compensant avec l'hypothèse de la transitivité de la préférence. Le dernier n'est pas toujours vrai en réalité, dans le sens où l'option A peut être préférée à l'option B et l'option B peut être préférée à l'option C mais, néanmoins, l'option C peut être préférable à l'option A pour l'utilisateur.

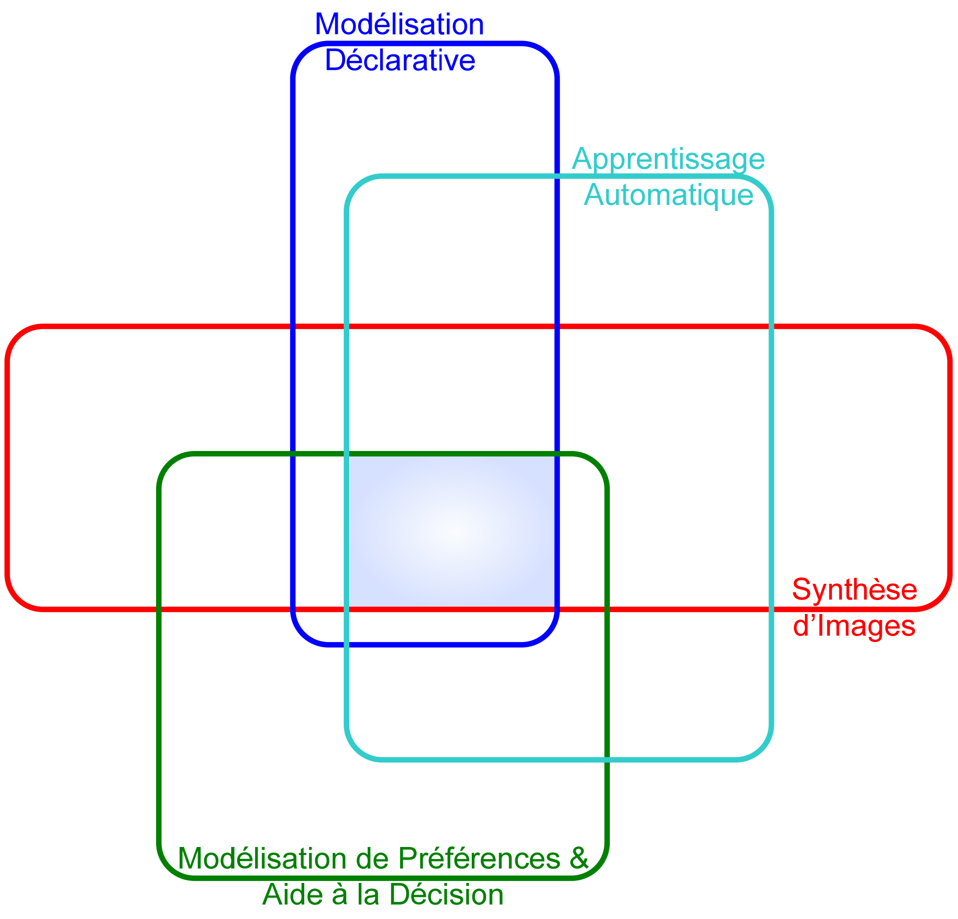


Figure 1 Domaines connexes à ce travail

Ce problème est l'un des soucis de la *modélisation de préférences*, un domaine qui vise à modeler la notion intuitive de la préférence humaine en ce qui concerne les options alternatives. Bien que proche de l'*analyse de décision*, ce domaine a ses propres fondements mathématiques robustes tout en incorporant des contributions de la science cognitive. Des définitions alternatives, concernant l'indifférence ou les degrés de préférence, ont été données afin de compenser des problèmes comme, par exemple, le célèbre problème du «sucre» : nous pouvons être indifférents entre deux cuillerées de sucre différant d'une graine seulement; cependant l'accumulation de cette indifférence, graine par graine, peut mener à la préférence claire en réalité.

L'apprentissage automatique est une région classique de l'intelligence artificielle consacrée à l'acquisition de la connaissance ou à la simulation du comportement basée sur une aide limitée sous forme d'exemples ou de règles. Il y a généralement deux catégories de méthodes, celles qui visent à acquérir la connaissance sous une forme compréhensible par les gens – l'approche de *boîte blanche* – et celles centrées principalement sur le comportement intelligent et moins sur la représentation de la connaissance intrinsèquement – l'approche de *boîte noire*. Puisque la plupart des méthodes dans ce domaine est fondée sur l'apprentissage à partir des exemples, même si un ensemble de règles a été prédéfini par des experts, des problèmes liés à la qualité et à la quantité de ces exemples surgissent. Un des problèmes dans ce domaine provient de l'incompatibilité potentielle entre la simplicité parfois implicite d'un ensemble fini d'exemples et la complexité grandissante de la fonction réelle ou du mécanisme à simuler. Un autre problème est celui de *surapprentissage,* où le mécanisme artificiel ne peut pas se comporter correctement quand de nouveaux exemples lui sont présentés. Dans ce cas-ci, le mécanisme est considéré comme incapable *de généraliser* sa connaissance à un degré acceptable.

Un exemple de généralisation incorrecte est présenté de façon vivante dans le puzzle logique célèbre qui concerne l'espion caché près d'une porte gardée essayant d'apprendre la combinaison du mot passe – réponse. L'histoire veut que, pendant que l'espion se cache tout près, un étranger arrive et demande d'entrer par la porte. Le garde crie *«six»* s'attendant à la réponse correcte du venu. L'étranger répond *«trois»* et le garde, satisfait de la réponse, le laisse entrer. Après un moment, une autre personne vient et cette fois-ci le garde crie *«huit»* auquel la personne répond *«quatre»*. Le garde est de nouveau satisfait de la réponse et laisse la personne entrer. Pendant ce temps, l'espion pense qu'il a pris conscience du mécanisme pour fournir la réponse correcte au garde: *c'est toujours la moitié du nombre prononcé par la garde*. Par conséquent, il se montre et demande d'entrer et cette fois-ci le garde crie *«dix»*. L'espion répond avec confiance *«cinq»* se trouvant immédiatement arrêté. *«La réponse correcte était «trois»* dit le garde comme l'espion est emporté. Évidemment la *généralisation* de l'espion, bien que conforme aux exemples qu'il a eus à sa disposition, n'a pas été celle correspondant à la réalité. La règle utilisée par le garde est vraiment simple mais différente et le but du puzzle lui-même est de trouver cette règle fondé sur tous les trois exemples – un avantage qui n'a jamais été offert à l'espion malheureux. Néanmoins, même la règle du garde n'est pas la seule conforme avec les exemples disponibles, laissant du champ, par conséquent, pour des généralisations alternatives.

Organisation de la thèse

L'organisation des chapitres restants suit une structure similaire. En particulier, le deuxième chapitre traite de la modélisation déclarative et des applications appropriées, en mettant l'accent sur l'environnement déclaratif de conception Open–MultiCAD parmi ces derniers, qui a servi de banc d'essai de la méthodologie proposée dans ce travail. Le troisième chapitre présente les bases de la modélisation de préférences, en se basant sur les notions appliquées dans le contexte actuel, et discute en détail les méthodologies d'aide à la décision à multi-critères utilisées dans ce travail, ainsi que les solutions alternatives les plus importantes. Le quatrième chapitre aborde l'apprentissage automatique, discutant des notions fondamentales et des algorithmes les plus significatifs du domaine ainsi que de leur applicabilité au contexte actuel. Le cinquième chapitre présente en détail le Module Intelligent de Profil d'Utilisateur proposé, décrit ses composants et aborde quelques questions d'implémentation. Le sixième chapitre décrit les expériences qui ont été menées afin d'ajuster et d'évaluer ultérieurement le prototype du Module Intelligent de Profil d'Utilisateur. Le dernier chapitre fournit les conclusions tirées de ce travail actuel et pose un certain nombre d'axes pour de recherche future.

1 Introduction

The current chapter aims to outline the motivation that has triggered this work, set the corresponding objectives and briefly discuss the context in which it has taken place as well as the contributing fields of the corresponding scientific areas.

We start with the suggestion that the problem we address here is a special case of a problem already existing in our everyday lives originating from the constantly and steeply improving performance of modern day computers and communications: that of information overflow. Next, we proceed to a brief presentation of the Declarative Modelling methodology, the needs it addresses and the inherent problems it exhibits. We state our goal and present a summary of the solution we have proposed and implemented within the scope of this work. In the remaining sections we provide an overview of the methodologies and corresponding research fields that have contributed to the solution.

1.1 Living in the Library

A large part of our everyday life benefits, directly or indirectly, from computer use and modern data communications. Schools and universities rely on this combination for basic tasks like text processing and presentations as well as for advanced research and distance learning. Businesses also perform research and development as well as executive decision making with their aid, while maintaining vast amounts of market, financial and customer data and performing remote management. Transportation relies on them for coordination and control, art has found new ways of expression, medicine is seriously supported in advanced tasks of diagnosis, therapy and drug design, it seems that computers and communications have spread into practically all aspects of human activity.

Of course, there is more than adequate reason for this penetration. Computers, on one hand, bear specific qualities to an extent far beyond human capacity. It is these qualities that actually make them one of the most versatile and attractive tools ever built. If we tried to isolate the core features upon which these qualities are built we would observe that computers, essentially, offer exceptional *speed* and *reliability* when it comes to performing data computations and data storage. Modern data communications, on the other hand, offer the ability to transfer in a fast and efficient manner, often in real-time, large volumes of information in the form of images, sounds, video and documents thus enhancing computers' abilities, making them practically ubiquitous in our everyday life.

Technological innovations during the last twenty years have seriously influenced the progress of computer industry and telecommunications thus dramatically improving the aforementioned qualities. Focusing on the former, hardware that the average user could only dream of during the eighties can now be found on every desk. A typical personal computer configuration of the middle-eighties included a CPU operating at 4.77 MHz handling roughly 1 million floating-point operations per second (1 MFLOP), 640 KB of RAM, and a floppy disk drive for 5.25" disks of 360 KB capacity, accompanied by a monochrome CRT monitor. At the same time, poor data communications could only offer voice and slow plain-text transfer to the average user. These features, although objectively not negligible, were practically not adequate for large scale applications. Today, a similarly priced workstation features a CPU operating at 3.4 GHz, 1 GB of RAM and a USB 2.0 port supporting flash memory sticks of 256 MB capacity or more, typically accompanied by a TFT colour monitor able to display a few billion colours. Moreover, transfer rates of 512Kbps and user friendly interfaces make data transfer of any form fast and easy. These numbers imply that contemporary machines represent an improvement of roughly 1000 ratio on every performance aspect – notice, for example, the aforementioned CPU frequencies – and, even more impressive, they are comparable with multi-million dollar supercomputers of the eighties as the famous Cray II. As a result of these advances, the computer user population has steeply increased. Old needs have transformed into new user demands that have to be fulfilled by these powerful machines and telecommunication channels. Nevertheless, as it is almost always the case, it takes more than raw computing power and transfer speed to fulfil these needs in an efficient manner.

J.L.Borges, in his excellent short story called *Library of Babel,* presents a metaphor that bears a lot of similarities with the modern day information era. The story is part of a collection called *Labyrinths* [Borges92] and was first published back in the 1940's, yet it appears to be remarkably predictive of the future. The location is the interior of a huge building of complicated architecture that resembles a labyrinth. The rooms of this building are connected through narrow corridors or staircases and the walls of every room are covered by shelves filled with books. The story is, in fact, revolving around these books: each book contains a *random* sequence of letters spanning from its first to its last page and, as it is only natural, most of the times the entire book does not make any sense. People, who live and act in the library throughout their lives, desperately seek a certain unique book that is assumed to contain the explanation of all other books as well as of the existence of the library itself. Sometimes, during their search, they stumble upon a meaningful word or phrase that appears among the otherwise incomprehensible pages of a book. As a result, they develop different ways of thinking based on these small fragments of randomly positioned reason. An impressive aspect of the story is the implication that some of the most important books that have ever been written or that *will be* written can also be found in the library simply because, after all, they are just sequences of letters. However, due to the miniscule fraction they represent against all possible letter sequences, these really important books are extremely hard to discover. Moreover, we are left with the feeling that people who have been living among random, incomprehensible pages for years, might not be able to appreciate and take advantage of these truly important books even if they discovered them.

The analogy with today's information overflow, although not intentional, is striking. Due to the aforementioned technological advances, we are surrounded by vast amounts of information, available under our fingertips or a few steps away in the form of websites, books, newspapers, magazines, TV shows, multimedia titles or publications. Nevertheless, we have to exercise our judgement almost constantly, against diverse media and approaches, in order to be able to filter the incoming stream of information and transform it to an acceptable subset of knowledge or entertainment not only in terms of size but also in terms of quality. Most of the times, this selection is not – and can not be – objective: we usually *filter in* that part of the incoming information that satisfies our needs at the time, ignoring parts that could be interesting for others. One way or the other, instead of selecting the best, we merely select what we consider as the best *for us,* tending to filter in that part of the input that is closer to our *preferences*.

1.2 Declarative Modelling and Early Phase Design

Declarative Modelling is a powerful methodology aiming to relieve the designer from the tedious task of explicitly stating the geometric properties of an object (or an environment) during early phase design. At this stage, the designer is more interested in ideas and functional requirements, seeking intuition coupled with practical use. Hence, Declarative Modelling offers the designer the ability to describe an object through a set of ambiguous properties and relations that could be interpreted in more than one ways with respect to the final geometric outcome. This versatility allows the designer to inspect alternative objects, all complying with the abstract description, and eventually choose that or those closer to his/her intuition.

However, there is a price to pay since the advantage of versatility and freedom during object design is translated to a computational disadvantage during solution generation and visualisation. Not all objects that conform to the submitted description are equally interesting. The alternative versions may be thousands and the designer is not always able or willing to inspect all of them in order to pick the most preferable designs. Several approaches have been proposed to address this problem based on neural networks, genetic algorithms, additional user feedback, example geometric representations or combinations of these mechanisms.

1.3 The Objective

The main objective of the current work is: *to provide a mechanism able to select and present to the user the subset of the solutions corresponding to a submitted declarative description closer to the specific user's preferences.*

1.4 The Proposed Methodology

In order to accomplish the aforementioned objective, we have proposed and implemented an Intelligent User Profile Module in the context of the Open-MultiCAD declarative design environment. This module is responsible for acquiring and maintaining user's preferences in a dedicated database by applying two alternative mechanisms:

- A transparent *Machine Learning* mechanism, trained by actual user selections during regular system use.
- A *Decision Support* mechanism, explicitly initialised by the user to reflect his/her personal preferences.

These preferences are then invoked whenever a declarative description is submitted by the user and processed by the system, thus presenting only the most interesting, i.e. preferable, subset of the generated solutions population.

1.5 Contributing Areas

The current work and the corresponding prototype combine notions from a wide range of fields. A visual representation of the interleaving of these fields appears in Figure 1.1.

Decision Analysis is one example of a field where the Decision Maker's preferences and expertise have to be translated into a meaningful model that will allow the selection of the most appropriate or preferable of a number of alternative options. Several mechanisms have been developed to facilitate this transfer from intuition to the corresponding concrete representation. The trade-off researchers usually have to resolve in this area is that between adequate user feedback and minimal user overhead in terms of time and complexity regarding the acquisition of this feedback. For example, there exist methods that require pair-wise comparisons of all alternatives with respect to each attribute affecting the overall selection. Pair-wise comparisons are one of the most widely accepted and straightforward methods for obtaining user preferences, since their adoption contributes to simplicity. On the other hand, albeit simple and straightforward for the user, these methods become impractical when more than a few tens of alternative options need to be evaluated since the number of pairs to be compared becomes prohibitive in terms of time.

Some methods relax that need for pair-wise comparison of alternative options compensating for it with the assumption of transitivity of preference. The latter is not necessarily true in real world in the sense that option A may be preferred over option B and option B preferred over option C but, nevertheless, option C may be preferable over option A for the user.

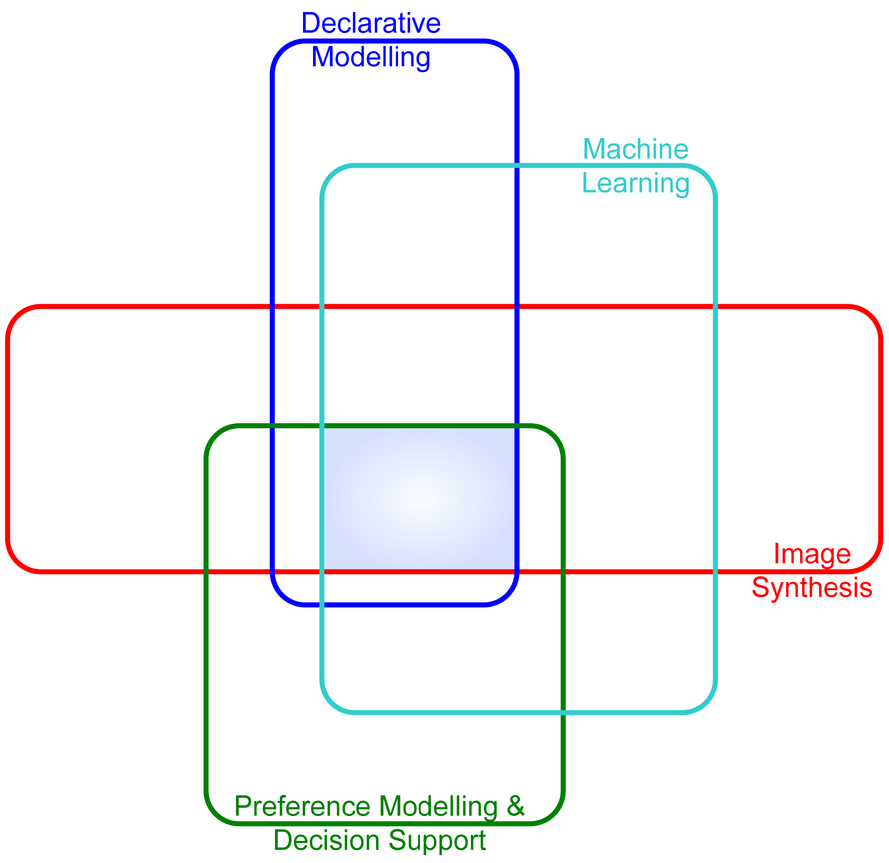


Figure 1.1 Fields contributing to this work

This issue is one of the concerns of *Preference Modelling*, a field that aims to model the intuitive notion of human preference with respect to alternative options. Although relative to Decision Analysis, this field has its own sound mathematical foundations while incorporating contributions from cognitive science. Alternative definitions regarding indifference or degrees of preference have been given in order to compensate for problems like the famous "sugar" example: we may be indifferent between two spoonfuls of sugar differing by only one grain; however the accumulation of this indifference, grain by grain, may lead to clear preference in reality.

Machine Learning is a classical field of Artificial Intelligence dedicated to the acquisition of knowledge or the simulation of knowledgeable behaviour based on limited guidance in the form of examples or rules. There are generally two categories of machine learning methods, those aiming to acquire knowledge in a form understandable by humans – the *white box* approach – and those concentrating mainly on intelligent behaviour and less on the knowledge representation per se – the *black box* approach. Since most methods in this area are based on learning by examples, even if a set of rules has been pre-defined by experts, problems connected with the quality and quantity of these examples arise. One of the problems in this field stems from the potential incompatibility between the simplicity sometimes implied by a finite set of examples and the increased complexity of the actual function or mechanism to be simulated. Another problem is that of *overfitting* to the examples, where the artificial mechanism is unable to behave correctly when new examples are presented to it. In this case, the mechanism is considered to be unable to *generalise* its knowledge to an acceptable degree.

An example of erroneous generalisation is vividly presented in the famous logical puzzle about the spy lurking by a guarded gate trying to learn the catchword-watchword combination. The story wants it that, while the spy lurks nearby, a stranger arrives and asks to enter the gate. The guard shouts *"six"* expecting the correct answer by the newcomer. The stranger replies *"three"* and the guard, satisfied by the answer, lets him in. After a while, another person comes along and this time the guard shouts *"twelve"* in which case the person replies *"six"*. The guard is once again satisfied by the answer and lets the person in. By this time, the spy thinks he has realised the mechanism to supply the correct answer to the guard: *it is always half the number uttered by the guard*. Hence, he shows up and asks to enter and this time the guard shouts *"eight"*. The spy confidently answers *"four"* only to find himself instantly arrested. *"The correct answer was 'five'"* says the guard as the spy is taken away. Obviously the spy's *generalisation,* although compliant with the examples he had available, has not been the one corresponding to reality. The rule used by the guard is really simple but different and the purpose of the puzzle itself is to find this rule based on all three examples – an advantage that was never offered to the unlucky spy. Nevertheless, even the guard's rule is obviously not the only one fitting the available examples hence leaving room for alternative generalisations.

1.6 Thesis Organisation

The organisation of the remaining chapters follows a similar structure. In particular, the second chapter elaborates on Declarative Modelling and the relevant applications, focusing on the Open-MultiCAD declarative design environment among the latter, which has served as the test bed of the herein proposed methodology. The third chapter presents the fundamentals of Preference Modelling, focusing on the notions applied in the current context, and discusses in detail the Multicriteria Decision Support methodologies employed in the current work as well as the most important alternatives. The fourth chapter addresses Machine Learning, discussing the basic notions and the most significant algorithms of the area as well as their applicability to the current context. The fifth chapter presents the proposed Intelligent User Profile Module in detail, describes the comprising components and addresses some implementation issues. The sixth chapter demonstrates the experiments that were conducted in order to adjust and subsequently evaluate the prototype Intelligent User Profile Module. The last chapter provides the conclusions drawn from the current work and sets a number of future work axes.

Part I

2 Declarative Modelling

Declarative Modelling is a powerful methodology for early phase design. Unlike *Imperative Modelling,* where all geometric details are explicitly defined by the designer, Declarative Modelling offers the ability to describe an object or an environment through a set of ambiguous terms. Different interpretations are then possible, leading to alternative final outcomes all based on the same abstract description. The current chapter commences with the origins and the rationale of the declarative modelling methodology. Next, Declarative Modelling by Hierarchical Decomposition is discussed focusing on its main differences when compared to the original concept. The most important software applications supporting declarative modelling design principles are presented. Among the latter, special emphasis is given to the Open-MultiCAD design environment that forms the context for the prototype that was designed and implemented within the scope of the current work.

2.1 Declarative Modelling Concept

During the early stages of the design process, the designer is usually not fully aware of the exact geometric properties of the object or environment to be created. The focus, at this stage, is on abstract characteristics rather than on concrete geometric properties. A typical example is a house, for which the following description appears to be quite natural and straightforward:

"It is a large house with two bedrooms and a rather spacious living room. The kitchen is not very big, but it is adjacent to the living room and communicates with it through a pass on their common wall. The bathroom is quite roomy and conveniently placed near the

bedrooms. The bedrooms are placed next to each other. The WC is near the living room which places all the public spaces on the east part of the house."

However, there are more than one ways to describe a house. The description for the same house could be as follows:

"The house is 10m wide and 15m long. The kitchen is 5m by 4m and the living room is 10m by 4m. One bedroom is 2.5m wide and 5m long whereas the other is 5m by 5m. The bathroom is 4m by 4m. Starting from the bottom left of the top view, the arrangement of the rooms is Bedroom, Kitchen, Living Room whereas from the top left it is Bedroom, Bath, WC and the same Living Room."

Notice that, in order to make the difference sharper, we have provided a geometric description as an alternative, avoiding any intuition regarding the house, concentrating on pure geometric properties. It is obvious that the former of the two descriptions is closer to the human way of thinking with respect to object or environment design. However, it is the latter kind of description, non-intuitive and purely geometric, that has to be specified by the designer in order to comply with the requirements of most commercial Computer Aided Design applications. In particular, in order to provide the visualisation of an object, CAD software requires a high degree of precision regarding the object's geometric properties. This, in turn, implies that by the time the object is visualised the designer has already performed the most interesting task without any assistance from the system: the task of object design. In other words, the CAD application can not be used to assist in idea generation but only to visualise ideas already conceived by the designer. This lack of support during the creative stage of the design process has led to the concept of declarative modelling and the corresponding modelling tools.

2.2 Declarative Modelling Design Process

In the Declarative Modelling context the *scene* is the object or environment to be designed. This scene may be described by declarative means, yielding the *declarative description* of the scene, or through a geometric representation – yielding a *solution* for the specific declarative description. The Declarative Modelling Design Process comprises three distinct stages covering the transition from the abstract declarative description of the scene to the resulting visualised solutions [Lucas89], [Plemenos95], namely:

- **Scene Description.** The definition of the declarative description of the scene.
- **Scene Generation.** Generation of geometric representations of the scene.
- **Scene Understanding.** Visualisation and information acquisition from the visualised solutions.

Each stage encompasses more than one steps of the overall procedure. In particular, the designer initiates the cycle by submitting a declarative description of the scene. This description integrates a mixture of design ideas, functional requirements, intuition, etc. As soon as the declarative description has been submitted one or more solutions complying with the description are generated and visualised by the corresponding mechanism. Subsequently, the designer is able to gain enhanced insight regarding the scene, enriched with ideas that may have not been obvious during the definition. Knowledge acquisition during scene understanding may be based simply on the designer's intuition or may be aided by the system. This is a topic of on-going research part of which is the current work as well as [Barral00]. The designer is then able to refine the declarative description in order to guide the solution generation to a more desirable result. Notice that the aforementioned stages actually form a cycle that may be repeated many times until the designer is satisfied with the final outcome. The interconnection of the three stages and the information exchange is shown in Figure 2.1.

According to the techniques applied for each of the aforementioned stages and their comprising steps, declarative modelling approaches can be distinguished in several subcategories. The main distinction is usually set regarding the *mode* of operation with respect to the generated solutions according to the corresponding declarative description input, namely:

- **Exploration mode.** The designer submits a declarative description and expects exploration of the *entire* solution space that fulfils this description. His/her purpose is to obtain novel or alternative ideas that conform to the submitted set of rules.
- **Solution Search mode.** The designer submits a declarative description and expects a *single* solution as a result. The task of narrowing down the range of the produced outcome to a single solution may be aided by additional feedback on behalf of the designer or restrictive interpretation of the submitted description.

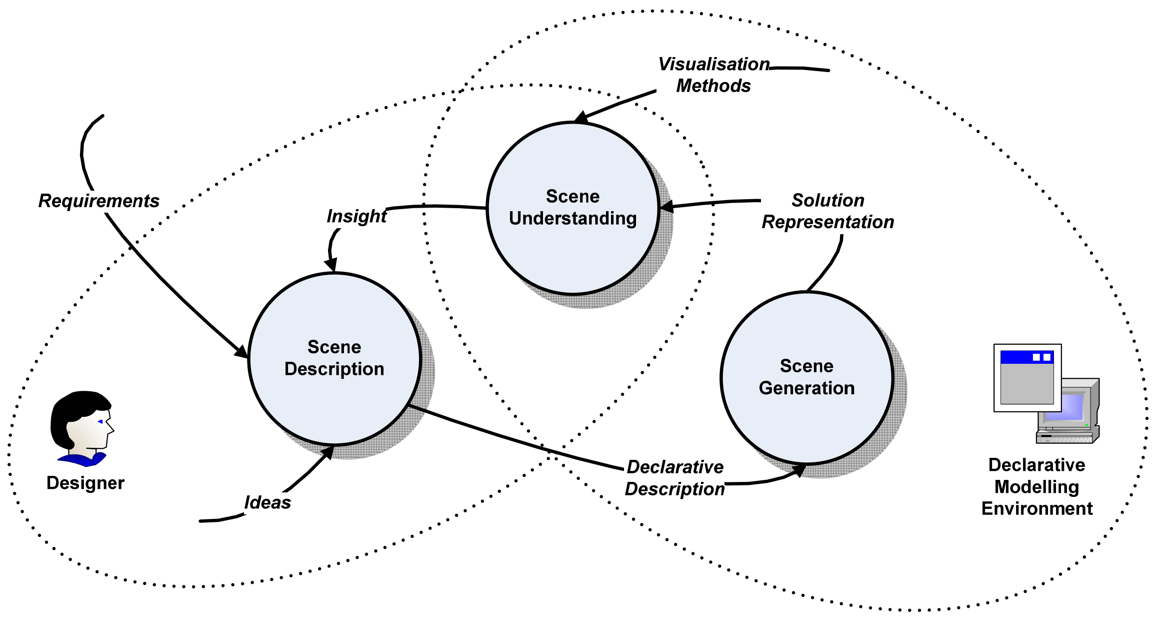


Figure 2.1 Declarative Modelling Design Process

One may argue that the Solution Search mode should be considered a special case of the Exploration mode since a single solution may always be picked among the solution space produced by the latter. Although formally this is true, the Solution Search mode usually incorporates the notion of *optimisation* in the sense that one of the *best* solutions complying with the submitted description is quested in contrast with the less strict approach inherent to the Exploration mode regarding the quality of the solutions. On the other hand, techniques to improve the quality of the solutions by concentrating on the most interesting solution sub-spaces have also been presented with regard to the Exploration mode in order to overcome time and space problems caused by the combinatorial explosion inherent to the task. These techniques are further discussed in the next sections presenting the most important Declarative Modelling tools.

2.3 Declarative Modelling by Hierarchical Decomposition

Although powerful and flexible enough to handle the definition of a wide range of objects or environments, the declarative modelling methodology may lead to complicated descriptions when dealing with complex scenes containing numerous objects and relations among them. Declarative Modelling by Hierarchical Decomposition (DMHD) [Plemenos95] attempts to reduce this complexity by separating the overall description into a set of simple descriptions of *sub-scenes*, each appropriately positioned in a *hierarchy* representing the complete scene. Following from this, it becomes apparent that DMHD may be applied only when two fundamental characteristics for the scene (object or environment) to be modelled are both present:

- **Increased Complexity.** The scene's complexity prohibits its modelling at a single level of decomposition.
- **Structural Knowledge.** The scene's structure is known to an extent that allows its recursive decomposition as a set of sub-scenes of reduced complexity.

In a sense, DMHD addresses the problem posed by the former of the two aforementioned characteristics, by taking advantage of the latter. This top-down approach of declarative design can be formally defined by the following set of recursive rules, adapted from the rules appearing in [Bonnefoi04]:

```
DMHD(currentscene)

    if (currentscene can be described by a small set of pre-defined high level
        declarative relations and properties)
        create the corresponding declarative description for currentscene
    else
        create a declarative description for currentscene
        using subscenes wherever needed to reduce complexity
        for each subscene used
            DMHD(subscene)
```

Algorithm 2.1 The Declarative Modelling by Hierarchical Decomposition technique

2.4 Declarative Description by Hierarchical Decomposition

The declarative description methodology, regardless of the explicit language or tool used for its representation, comprises a set of basic elements used to express the semantics of the description and serve as rules for the solution generation process. In particular, each scene described in a declarative manner usually consists of the following:

- A set of declarative objects
- A set of declarative properties required for or exhibited by these objects
- A set of declarative relations among these objects

In the case of Declarative Modelling by Hierarchical Decomposition the objects may well be sub-scenes that are recursively described in the same manner. In the following we concentrate on the declarative description of physical, visually representable, objects. Hence, the objects, relations and properties that are mentioned, are connected with morphological and spatial elements. In particular, in order to clarify the aforementioned declarative description notions we consider an example of a typical habitation. The declarative objects of the corresponding scene appear in Table 2.1.

Declarative Objects	Public Zone Private Zone Children's Bedroom Parents' Bedroom Bathroom Kitchen Living-Room WC Corridor

Table 2.1 Declarative Objects of Habitation scene

The designer may require each one of these objects to demonstrate certain abstract properties. An example set of declarative properties for the objects of Table 2.1 appears in Table 2.2.

Declarative Properties	Corridor is long and narrow Parents' Bedroom is big and square shaped

Table 2.2 Declarative Properties of the Habitation scene objects

Declarative properties are abstract and closer to the human way of thinking in the sense that, although they are connected with an object's morphology, they allow for alternative, yet all valid, interpretations of it. The explicit range of these interpretations is, up to an extent, connected with the specific tool used for solution generation in a given context.

The relations that may appear in a declarative description of a scene fall into three categories [Miaoulis02] according to the kind of connection they represent between the related objects. These categories refer to binary relations but they may be trivially extended to include relations with an arbitrary number of members.

- **Meronymic.** One of the two objects forms part of the other. It is worth noticing that, although in the present context we focus on visually representable objects, this *part-of* category of relations does not necessarily imply physical interconnection between the objects. In a wider sense, meronymic relations may support the notion of *containment* among *logical objects* rather than strictly physical.
- **Correlational.** One (or more) of the properties of the two objects are connected by means of comparison. Similarly to the previous category, in a broader context, the relation may refer to physical properties like length, colour, texture, as well as logical properties under the condition that the latter have been concretely defined and, thus, they are comparable.
- **Spatial.** The positions of the two objects are comparatively related. Typically, the category of *Spatial* relations may be considered as a special case of the *Correlational* category, in the sense that the relative position of an object with respect to another may be defined as a set of rules regarding its geometric properties specifying its position and size. However, this translation is, generally, not trivial and, at the declarative description level, we deal with the relations in a form closer to the human way of thinking. In other words, a simple spatial relation seems more natural and closer to the declarative modelling rationale than its equivalent set of correlational relations. This distinction will become more evident during the discussion of the Open-MultiCAD environment, in subsequent sections. A set of declarative relations for the running example appears in Table 2.3.

Declarative Relations	***Meronymic***	Public Zone is part of the Habitation Private Zone is part of the Habitation Corridor is part of the Habitation Children's Bedroom is part of the Private Zone Parents' Bedroom is part of the Private Zone Bathroom is part of the Private Zone Kitchen is part of the Private Zone Living-Room is part of the Public Zone WC is part of the Public Zone
	Correlational	Bathroom is larger than the WC Living-Room is larger than the Kitchen
	Spatial	Bathroom is near Children's Bedroom Parents' Bedroom is adjacent to Children's Bedroom south WC is near Living-Room Public Zone is adjacent to the Private Zone on its east

Table 2.3 Declarative Relations among the Habitation scene objects

Notice that in the specific description we have applied Declarative Modelling by Hierarchical Decomposition since the *Public Zone* and *Private Zone* objects are actually sub-scenes, *containing* some of the subsequent objects.

The description assembled by Table 2.1, Table 2.2 and Table 2.3, although complete and accurate in terms of content, fails to provide an *intuitive* overview of the declarative aspects of the scene. In a sense, although the visual interpretation of the actual geometric representation of the scene is not desired at this level, visual information regarding the declarative description itself, in a more intuitive manner, is welcome. Hence, an enriched tree structure forms a very common alternative method for the representation of a declarative description. Figure 2.2 presents a graphical equivalent of the Habitation scene, based on the guidelines proposed in [Miaoulis02]. At this point, it is worth noting that, for the sake of uniformity, we have based the Habitation scene example on the verbal house description given at the beginning of this chapter.

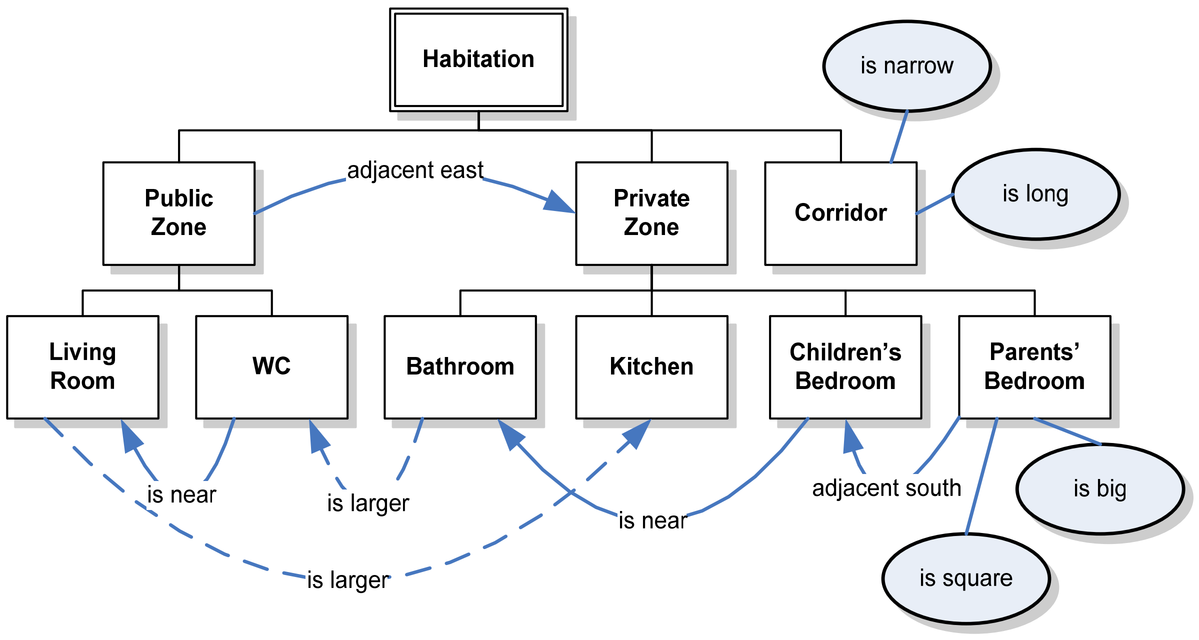


Figure 2.2 Graphical Representation of the Declarative Description of the Habitation scene

2.5 Declarative Modelling Tools

Since the introduction of declarative modelling methodology a number of efforts have taken place in the form of applications applying its principles, referred as *Declarative Modellers*. We may generally distinguish two kinds of modellers depending on their goal in regard to the design procedure. In particular, we have

- **Dedicated Modellers.** Modellers falling into this category concentrate on a specific kind of object or environment to be designed. The advantage of this approach is the fact that the corresponding applications often provide additional tools, specialised in the treatment of a particular type of scenes, further facilitating the designer's work.

- **General Purpose Modellers.** The modellers of this category aim to handle a wide variety of objects or environments without focusing on a specific paradigm or domain. This forms their main advantage, due to their flexibility and their ability to adapt to the needs of the designer regardless of the specifics of the scene to be processed.

In the following, we provide a brief description of projects and applications that have employed the principles of declarative modelling in various fields and to different extents. We dedicate a large part of the remaining part of this chapter to the Open-MultiCAD Declarative Design Environment since it incorporates part of the preliminary stages of the current work and has also served as the basis for its main part, the Intelligent User Profile Module, which is described in the next Part of this thesis.

2.5.1 Dedicated Modellers

2.5.1.1 VoluFormes

This dedicated modeller [Chauvat94] consists of two modules:

- **VoluBoites.** This module is responsible for generating alternative placements of bounding boxes in a 3D space. The arrangements are produced serially and they are subject to user approval or rejection. Once the user's approval is available the operation of the next module is taking place. VoluBoites applies a declarative modelling methodology for the description of the placement of the control boxes.
- **VoluScenes.** This module is taking advantage of the placement of the control boxes decided in the previous module in order to create forms based on growth mechanisms.

The output of the two modules, as presented in [Plemenos04], appears in Figure 2.3.

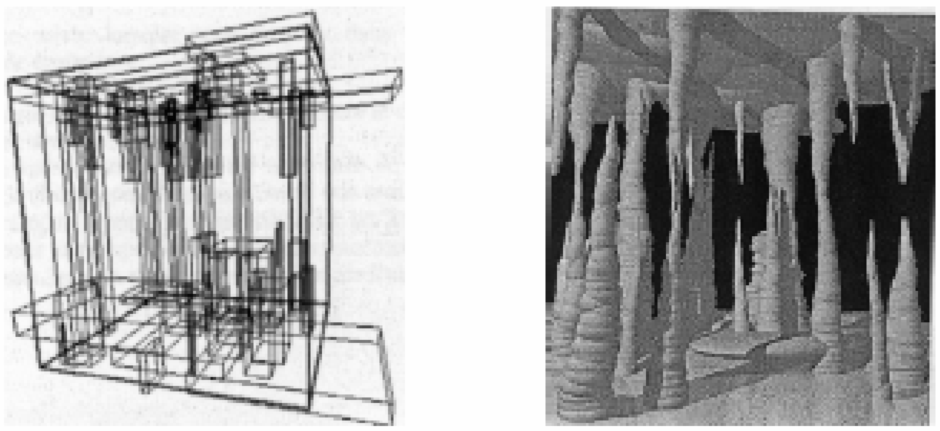


Figure 2.3 Bounding box arrangement (left) and the corresponding final scene (right)

2.5.1.2 DEM^{2}ONS

This project covers the entire range between the abstract conceptual form of an object or environment and its visual representation [Gaildrat03]. It has launched several sub-projects implementing alternative methods for the exploration of the solution space, concentrating on the domains of Architecture and Interior Design. Table 2.4 summarises the alternative methods used in the context of these sub-projects for the resolution of the constraints comprising the declarative description.

Algebraic	Approximative
	Exact
Deductive	Expert Systems
	Re-writing Systems
Constructive	Descending
	Ascending
Local Propagation	Propagation of Conflicts
	Propagation of Degrees of Freedom (DOF)
CSP	Hierarchical
	Dynamic
	Numerical
Metaheuristics	Local Search
	Genetic Algorithms

Table 2.4 Constraint Resolution Methods Applied in DEM^{2}ONS Projects [Gaildrat03]

An example of the output of the DEM^{2}ONS_GA tool [Sanchez00], based on genetic algorithms for solution generation, appears in Figure 2.4.

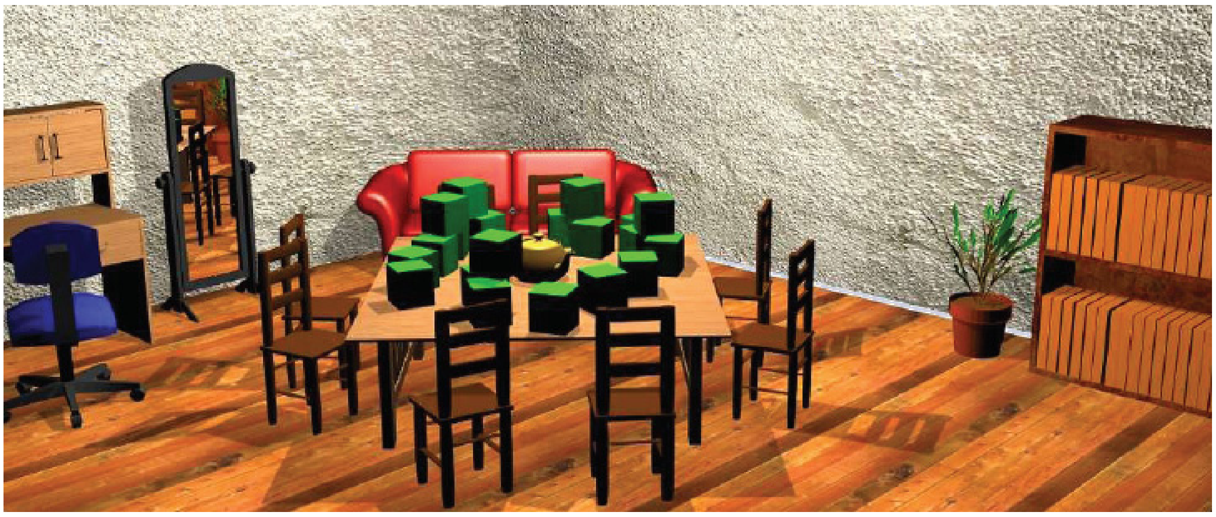


Figure 2.4 Part of scene produced by DEM^{2}ONS as it appears in [Gaildrat03]

2.5.1.3 PolyFormes

PolyFormes comprises a tool for the generation of regular or semi-regular polyhedra based on a declarative description [Martin99]. This description may correspond to one or more geometric representations in which case all solutions complying with it are provided by the system, when the number of solutions is finite. In the opposite case, where the description suggests an infinite number of alternative polyhedra, and this is possible with the specific description mechanism provided by the tool, the system stops at a certain depth the exploration of the solution space. An example semiregular polyhedron generated by the system appears in Figure 2.5.

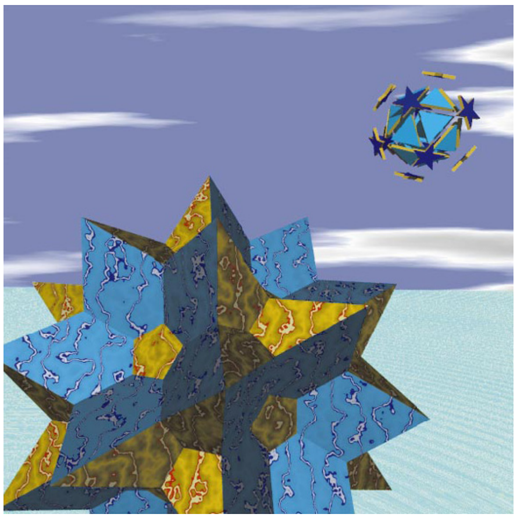


Figure 2.5 A semi-regular polyhedron (foreground) and its exploded version showing its constituents (background) [Martin99]

2.5.1.4 MégaFormes

This is a dedicated declarative modeller aiming to represent megalithic sites through the process of declarative arrangement of bounding boxes and subsequent replacement of them by rendered representations of the corresponding objects [Poulet96]. The approach is similar with the one used in VoluFormes in the sense that only the first part of the methodology used is explicitly based on declarative modelling. An example of the system's output, as shown in [Gaildrat05], appears in Figure 2.6.

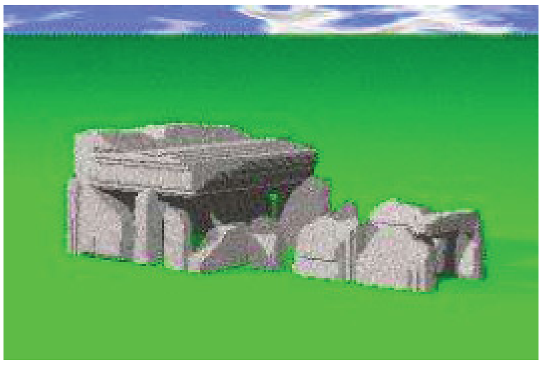


Figure 2.6 MégaFormes output as shown in [Gaildrat05]

2.5.1.5 BatiMan

This is a significant declarative modelling tool [Champciaux98a], [Champciaux98b] in the sense that it combines Constraint Satisfaction Problem (CSP) techniques for solution generation and Machine Learning mechanisms to assist the user during the scene understanding phase of the declarative modelling process to define the *target concept* which resembles the notion of user preferences of the current context. In particular, the tool models buildings based on a finite set of representative elements that are combined to provide the alternative solutions. It maintains a tree structure for the unsupervised classification of the solutions in *concepts* and offers the user the ability to examine the representative solutions for each class in order to create a subset of solutions describing the *target concept*. This is accomplished based on subsequent user evaluation of class representatives which have been selected based on the degree of class characteristics they share.

The machine learning component of this tool is based on the quite restrictive assumption that user preference, as reflected in the target concept, is *similar* for *similar solutions*. As a result, user's preferences are modelled based on the solution classes and not on the solutions themselves. In other words, by its construction, the tool prevents the user from approving solutions from alternative classes while, concurrently, rejecting solutions from the same classes.

2.5.2 General Purpose Modellers

2.5.2.1 WordsEye

This tool, currently undergoing beta testing, accepts a paragraph as input and generates an image interpretation of the text. It is based on pre-defined objects and properties and produces a single solution for the given description. Figure 2.7 presents the output created based on the following phrase:

"The couch is against the wood wall. The window is on the wall. The window is next to the couch. The door is 2 feet to the right of the window. The man is next to the couch. The animal wall is to the right of the wood wall. The animal wall is in front of the wood wall. The animal wall is facing left. The walls are on the huge floor. The zebra skin coffee table is two feet in front of the couch. The lamp is on the table. The floor is shiny."

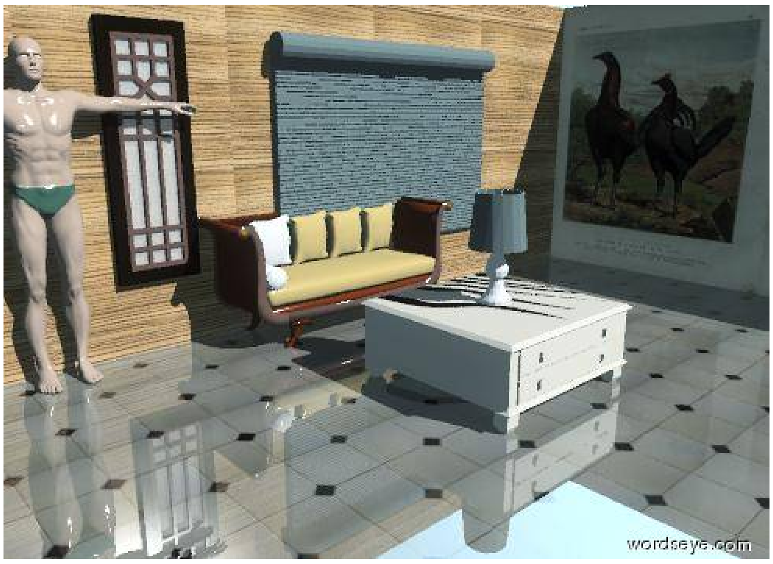


Figure 2.7 WordsEye output based on text scene description [SemanticLight06]

2.5.2.2 SpatioFormes

This general purpose modeller accepts a description defining the declarative properties of a scene as input based on a specific vocabulary and the corresponding syntax rules. The mechanism generates alternative geometric representations of the declarative description in the 3D space as voxel sets. The generation of the solution space takes place through the breadth-first exploration of the tree of alternative valid solution representations [Poulet94].

2.5.2.3 CAPS

Constraint-based Placement for Scene composition (CAPS) [Xu02] offers a realistic arrangement of objects in a three dimensional space. The arrangement is based on abstract user constraints combined with pseudo-physics for the calculation of position in terms of stability and friction. In addition, each object is enhanced by semantic content that affects its placement with respect to the other objects of the scene. The modeller provides a single arrangement considering all available information. Alternative arrangements may be obtained either through direct manipulation of an object's position or through the alteration of each object's probability to support (or rest) on another. Nevertheless, the modeller should generally be considered to operate in *solution search mode* rather than *exploration mode.* Figure 2.8 shows an example scene generated by CAPS.

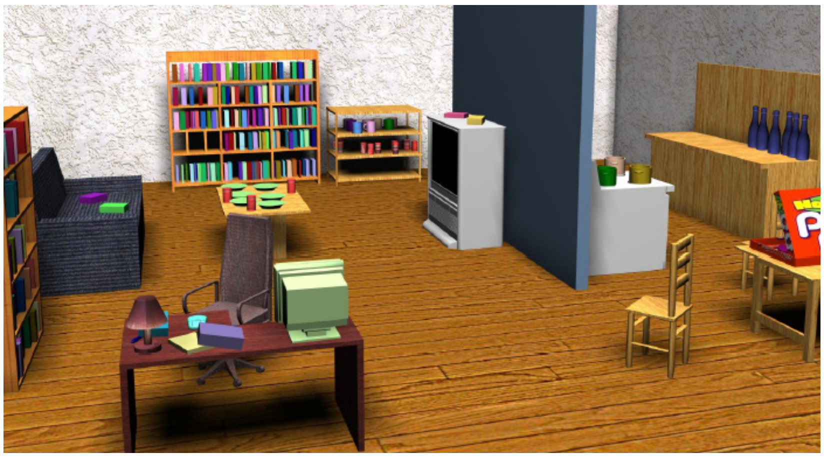


Figure 2.8 Object arrangement generated by the CAPS tool [Xu02]

2.5.2.4 MultiFormes

This is a general purpose declarative modeller that combines Constraint Satisfaction Problem resolution techniques with heuristic methods to accelerate solution generation [Plemenos91],[Ruchaud02] and has evolved through several versions. Each scene is described as a set of geometric objects and a set of relationships among these objects, expressed in a special declarative modelling language implementing Declarative Modelling by Hierarchical Decomposition. Relationships can be high-level (imprecise) as well as low-level (precise) and solution generation is geometry-based. Boxes and convex polygons are currently supported by the tool, sufficing to produce highly detailed results as the example appearing in Figure 2.9. Machine Learning techniques have been proposed in the context of this environment either in the form of detail reduction techniques [Boughanem94], through neural networks learning positions and dimensions of solution objects [Boughanem94] or declarative scene hierarchies [Plemenos02] as well as by using a genetic algorithm [Plemenos02] guided by the user to gradually concentrate on interesting solutions.

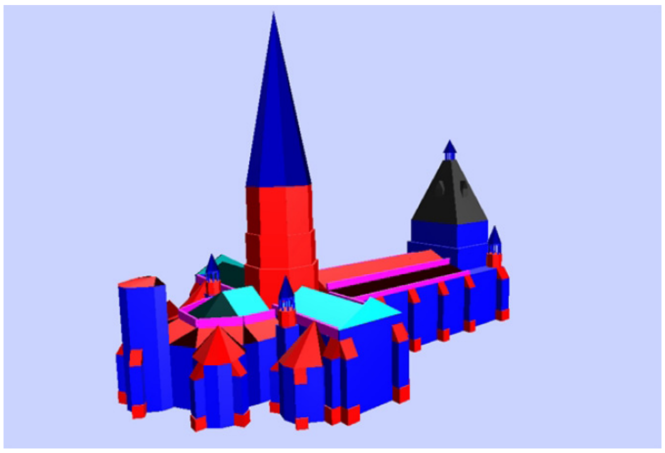


Figure 2.9 Example MultiFormes output of a Romanesque church description [Ruchaud02]

2.5.2.5 XMultiFormes

The specific project [Sellinger97],[Sellinger98] implements the Cooperative Computer Aided Design (CCAD) framework for generative modelling systems. It combines the MultiFormes declarative modelling tool with an imperative, interactive modeller allowing modification of the geometric scene representation. A key feature of the environment is the ability to reverse engineer a, possibly modified, geometric representation of a submitted declarative scene, thus extracting new or modified semantic information. The latter may be enriched with information originating from the original declarative description, the corresponding geometric representation or direct user feedback.

2.6 The Open-MultiCAD Environment

Open-MultiCAD is an intelligent information system, comprising an integrated design environment that supports the entire cycle of the Declarative Modelling methodology. The declarative description methodology supported by the environment applies the principles of Declarative Modelling by Hierarchical Decomposition (DMHD) [Plemenos95] whereas the main solution generator module is based on Constraint Satisfaction Problem resolution techniques and forms an evolution of the corresponding module of the MultiFormes environment [Plemenos91],[Ruchaud02]. The initial MultiCAD environment has been described in detail in [Miaoulis02]. The current state of the project, and the corresponding prototype, comprises tools for a number of tasks related with the transition from the abstract scene definition to its geometric counterpart and vice versa as well as with the understanding of the scene.

2.6.1 Modules

Research has taken place and corresponding results have been presented, within the context of the following tasks:

- Normalisation and refinement of the declarative description according to the rules governing a specific domain of application, e.g. Architecture [Makris03].
- Generation of solutions conforming to the declarative description through the use of genetic algorithms in order to focus on interesting subsets of the solution space [Vassilas03].
- Generation of the entire solution space conforming to the declarative description through the use of constraint satisfaction techniques [Bonnefoi02].
- The current work, applying intelligent selection of generated solutions based on previous evaluations and/or predefined rules using machine learning and decision analysis techniques initially presented in [Bardis04] and [Bardis06].
- Design and implementation of specialised repositories supporting the aforementioned efforts with the capability of incorporating of knowledge from a new domain with minimal effort [Ravani03].

- Reverse engineering of the geometric representation of a scene yielding the corresponding declarative description which, in turn, is used to generate solutions *similar* to the original [Golfinopoulos05].
- Modelling of Architectural Styles as a set of strict (*hard*) and flexible (*soft*) constraints. Application of the resulting style model, through genetic algorithms, to the solution generation in order to improve conformity of the generated solutions to a specific architectural style [Makris05].
- Construction of a collaborative design environment to support the simultaneous definition and processing of the declarative description by a team of designers. Support of a distributed architecture, allowing the designers to be at distant locations and communicate with the system as well as with the rest of the team using standard network protocols [Dragonas05].
- Implementation of tools featuring open architecture, being able to support alternative domains through the definition of the relevant declarative relations and properties in the supporting database. Minimal changes are required to the prototype itself in order to support alternative domains. This feature has been the result of contributions from all aforementioned works.

2.6.2 Open-MultiCAD Database

All information related to the declarative modelling cycle is stored and maintained in a specialised database. This repository has been designed and implemented in a manner that makes it capable of supporting and representing alternative domains and the corresponding declarative objects, properties and relations. The ER diagram of the MultiCAD database appears in Figure 2.10. Since this database incorporates all notions supported by the core Open-MultiCAD environment, we elaborate on it, explaining the role of all entities appearing therein.

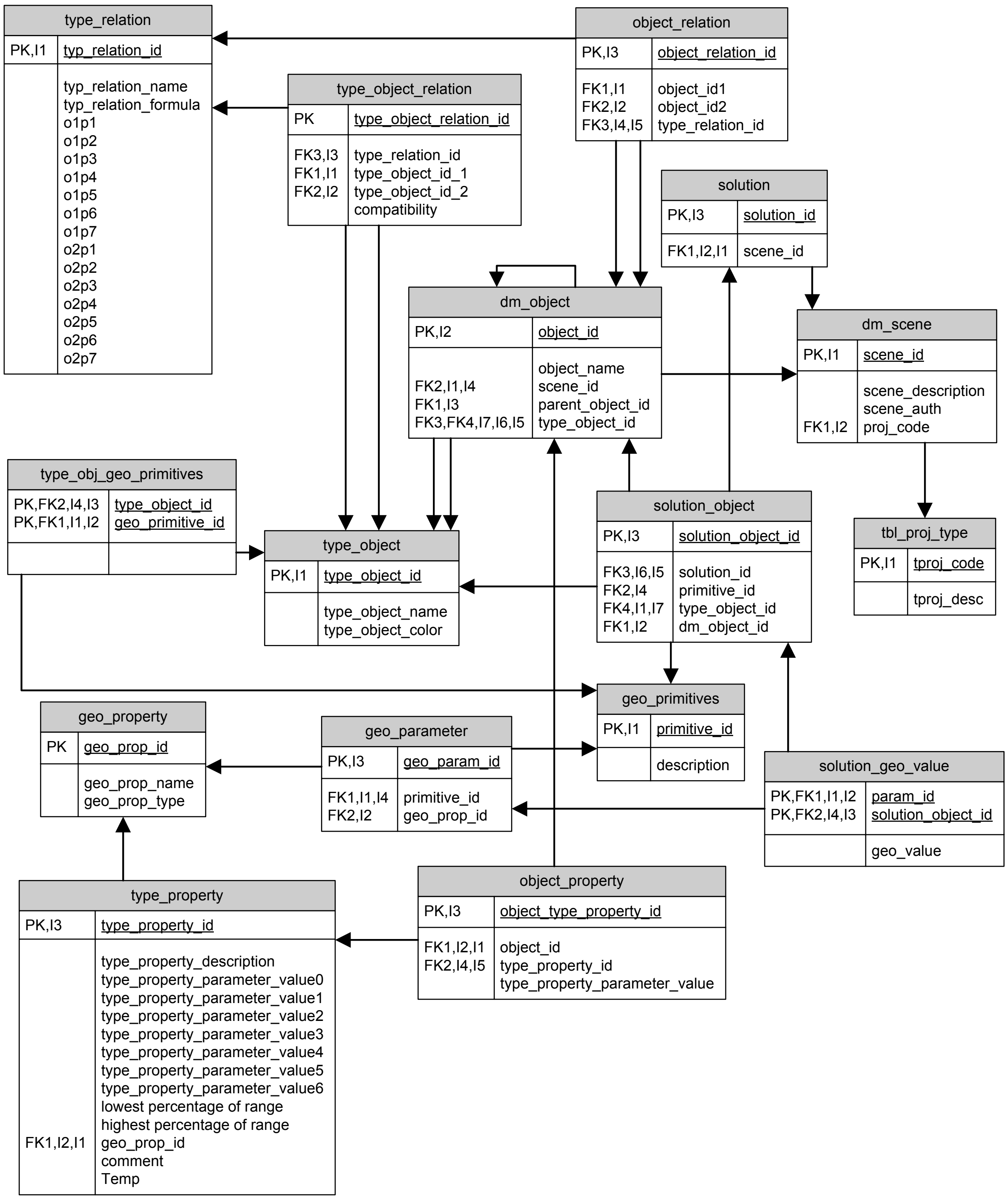


Figure 2.10 The Open-MultiCAD Database

In particular, each submitted scene is stored in the entity *dm_scene*, including information regarding the author, a small verbal description and the project type it corresponds. The project type represents the kind of building assembly the scene represents, e.g. *'habitation'*, *'office'*, *'hospital'*, etc. all stored in *tbl_project_type.* As already discussed in the previous sections, each scene contains declarative objects, having declarative properties and connected through declarative relations. These declarative objects are stored in *dm_object.* Each one of these objects may be part of a parent object, due to hierarchical decomposition, and is of a specific type. Object types are stored in *type_object* and some examples are *'kitchen', 'bedroom', 'living-room',* etc. In other words, the contents of *type_object* represent the classes of declarative objects available to instantiate and include in a declarative description. These classes are connected with one or more geometric primitives, e.g. *'parallelepiped'*, *'hemisphere'*, etc. These geometric primitive solids – stored in *geo_primitive* – are used during solution generation for the geometric interpretation of the declarative object. As an example, an object of the object type *'kitchen'* may be allowed to be represented only by a *'parallelepiped'* in the geometric representation whereas an object of the object type *'roof'* may be allowed to be represented by a *'hemisphere',* a *'prism',* etc. This connection is suggested in the *type_object_geo_primitives* entity that actually represents the implementation of a many-to-many relationship between the *type_object* and the *geo_primitive* entities. The *geo_parameter* entity relates the different geometric properties – stored collectively in *geo_property* – with different geometric primitive solids. For example, the *'hemisphere'* is only connected with the *'radius'*, the cylinder with the *'radius'* and the *'height'*, the parallelepiped with the *'length'*, the *'width'* and the *'height'*, etc. The entity *solution_object* contains the instances of all objects appearing in all solutions. Following from the previous discussion, each object participating in a solution corresponds to a geometric primitive, having certain properties of a specific value. The latter are stored in the separate entity *solution_geo_value.*

The declarative properties available are stored in the *type_property* entity. An example declarative property contained therein is the *'is long'* property that is translated to the higher portion of the overall range of the *length* geometric property, as suggested by the appropriate entity attribute *geo_prop_id*. A declarative property may also be combined with one of alternative modifiers, for example *'very'* thus giving a declarative property of the form *"is very long"*. The valid modifiers for each declarative property are stored in the *type_property_parameter_value* attributes of the *type_property* entity. The interpretation of the modifier is an equal portion of the sub-range corresponding to the specific declarative property. In order to clarify this connection between the declarative property and the modifier consider the example: assuming that the geometric interpretation of *length* for an object varies from 1 to 100, in case the object has been described as *is long,* its length will range from 70 to 100 in the corresponding solutions. This is suggested by *lowest_percentage_of_range* and *highest_percentage_of_range* being 70 and 100 respectively. Moreover, there are three modifiers for *'is long',* namely *'a little', 'medium', 'very',* thus yielding *'is a little long', 'is medium long'* and *'is very long'.* These are stored in *type_property_parameter_value0, type_property_parameter_value1* and *type_property_parameter_value2* respectively. The corresponding sub-range for *'is very long'* is then from 90 to 100. The aforementioned numbers are summarised in Table 2.5.

Valid Range for "length" geometric property (defined by the environment)			1..100
lowest percentage of range for "Is Long" declarative property (stored in database)			70%
highest percentage of range for "Is Long" declarative property (stored in database)			100%
Declarative Property	***Modifier***	***Final Declarative Property***	***Corresponding Range for geometric property "length"***
Is Long	**-**	***Is Long***	70..100
Is Long	***Very***	***Is Very Long***	90..100
Is Long	***Medium***	***Is Medium Long***	80..90
Is Long	***A Little***	***Is A Little Long***	70..80

Table 2.5 Declarative property and geometric interpretation example

Although the actual numerical interpretation of the declarative property and its modifiers may be argued – this is an abstract ambiguous property after all – this is an example of the care that has been taken to create an *open* system able to support new declarative properties or modifications to existing ones by just adding or modifying records to the appropriate database tables. This open architecture is further stressed by the representation of the declarative relations in the Open-MultiCAD database that is discussed next.

The declarative relations available are stored in a special format in the *type_relation* entity. Each relation connects the geometric properties of two objects however the objects themselves are not stored in the entity. This connection is accomplished via the *object_relation* entity. The *type_relation* entity contains geometric properties and their interconnection in a special formatting that can be translated and evaluated by the application performing the solution generation. In order to clarify this connection consider the excerpt of the *object_relation* table of the Open-MultiCAD database representing the *'attached to the south of'* declarative relation appearing in Table 2.6, shown in a vertical form while it really represents a record, i.e. a single row, of the table. The *typ_relation_formula* field contains the arithmetic formula that connects the geometric properties participating in the specific declarative description. The specific properties appear in the same record in a highly organised manner. In particular, the properties referring to the first of the two related objects can be found in the fields o1p1,…, o1p7 of the same record. Similarly, the properties that participate in the specific declarative relation and correspond to the second object are stored in the o2p1,…,o2p7 fields. The syntax of the arithmetic formula itself is quite straightforward and easily translated by the solution generator. As an example we translate the first term of the formula, when referring to the declarative relation

Parents' Bedroom is adjacent to Children's Bedroom south

that appears in Table 2.3. In this case, the parallelepiped representing the parents' bedroom is considered to be object1 (o1) and the parallelepiped representing the children's bedroom is object2 (o2). Therefore, the first term of the formula

(o2p1 <= o1p1 + o1p4)

can be translated as

the x coordinate of the children's bedroom must be less or equal to the sum of the x coordinate of the parents' bedroom and the parent's bedroom length

because `o2p1` contains `ox` (x-coordinate) while `o1p1` contains `ox` (x-coordinate) and `o1p4` contains `l` (length). x-coordinate and length are defined as such in the *geo_property* table.

Field Name	***Field Value***
typ_relation_id	`as`
typ_relation_name	`adjacent south`

typ_relation_formula	`( o2p1 <= o1p1 + o1p4 ) & ( o2p1 + o2p4 >= o1p1 ) &`
	`( o2p2 = o1p2 + o1p5 ) &`
	`( o2p3 + o2p6 >= o1p3 ) & ( o2p3 <= o1p3 + o1p6 )`

o1p1	`ox`

o1p2	oy
o1p3	oz
o1p4	l
o1p5	w
o1p6	h
o1p7	
o2p1	ox
o2p2	oy
o2p3	oz
o2p4	l
o2p5	w
o2p6	h
o2p7	

Table 2.6 Record from *type_relation* representing *'adjacent_south'* declarative relation

2.6.3 Open Architecture and Flexibility

It is important to notice that the overall approach of the Open-MultiCAD database is flexible enough to support declarative properties that are not necessarily connected with geometric properties. We could, for example, model the declarative relation

is darker than

by adding the appropriate records to the corresponding tables. In particular, the relation itself could be represented in the *type_relation* table as an additional record containing the values appearing in Table 2.7. The meaning here is that the *luminosity* of object 1 is less than the *luminosity* of object 2. The formula could be much more complicated, including more properties representing colours, shading, etc. The appropriate entries for these properties, as the property *luminosity* of the specific simplified example, will have to be added – if not already present – in the *geo_property* table containing all properties participating in the solution representation.

Field Name	*Field Value*
typ_relation_id	`dt`
typ_relation_name	`darker than`
typ_relation_formula	`( o1p1 < o2p1 )`
o1p1	`lum`
o1p2	
o1p3	
o1p4	
o1p5	
o1p6	
o1p7	
o2p1	`lum`
o2p2	
o2p3	
o2p4	
o2p5	
o2p6	
o2p7	

Table 2.7 Record in *type_relation* representing new declarative relation

It is also worth noting that the specific approach makes it possible to express declarative properties that involve more than one property of an object as special cases of declarative relations, i.e. unary relations. For example, the declarative property

is square shaped

could be expressed by the record shown in Table 2.8 in the *type_property* table.

Field Name	Field Value
typ_relation_id	sq
typ_relation_name	squared shaped
typ_relation_formula	(o1p1 = o1p2)
o1p1	l
o1p2	w
o1p3	
o1p4	
o1p5	
o1p6	
o1p7	
o2p1	
o2p2	
o2p3	
o2p4	
o2p5	
o2p6	
o2p7	

Table 2.8 Record in *type_relation* representing a special declarative property

In addition to the aforementioned capabilities, Open-MultiCAD is able to cooperate with alternative solution generators based on a concretely defined information exchange interface. In particular, each declarative description is translated to a set of arithmetic expressions and inequalities representing the constraints that have to be satisfied by the corresponding solutions. This set of constraints can be appropriately translated, with minimal effort, in order to be used in alternative Constraint Satisfaction tools as well as to be applied as a fitness function in the context of a genetic algorithm.

Moreover, the module supporting reverse engineering is capable of accepting geometric representations, based on a simple syntax, and yield a declarative description that complies with these representations. Geometric representations originating from various design environments can be translated to conform to the aforementioned syntax and subsequently be used in order to produce declarative descriptions based on examples.

A similar approach has been applied to the Intelligent User Profile Module, proposed and implemented within the context of the current work. In particular, our methodology has maintained the requirement for open architecture, thus concentrating on the ability to support input from multiple sources and the comparability of this input with respect to the user's preferences.

2.6.4 Solution Generation

A number of alternative approaches have been applied to the design and implementation of the solution generator module for the Open-MultiCAD environment. Nevertheless, despite the different methods used, these generators share a set of characteristics that are dictated by the declarative modelling methodology and its incorporation to the specific environment.

In particular, each object, either complex, i.e. containing other objects, or simple, i.e. a terminal node in the corresponding declarative hierarchy, is initially represented in the geometric scene interpretation as a bounding box. This implies that, in the case of complex objects, all children nodes are placed within the topological limits of the parent bounding box. As a result, there is a top-down interpretation of the declarative relations between objects. For example, if the public zone is requested to be *adjacent east* to the private zone, then requesting the living room, which belongs to the public zone, to be *adjacent west* to the kitchen, which belongs to the private zone, leads to a contradiction and, as a result, does not yield any solutions for the specific description. Nevertheless, a relation between two objects belonging to two different parent objects generally makes sense, since the user may explicitly require the living room to be *adjacent east* to the kitchen. Notice that the latter is not necessarily implied by the previous declarative relation between the public and private zone. In other words it may be the case that the public zone is adjacent east to the private zone and, at the same time, the living room is not adjacent east to the kitchen.

Based on the above discussion, in the case of constraint satisfaction solution generators, we may distinguish two separate sub-stages during solution generation. One relies entirely on the bounding boxes approach to generate the arrangement of the objects and the other incorporates specific geometric properties of the objects in order to produce the final geometric representation of the scene.

The alternative placements of the objects are computed based on the corresponding bounding box whereas the final geometric interpretation incorporates the explicit geometric properties of the solids used to represent it. This distinction between the declarative description and the final geometric interpretation is also evident in the MultiCAD database, where all information covering the entire range of the declarative methodology is stored. Figure 2.11 shows a few example alternative bounding box arrangements for an *igloo* habitation scene, based on the declarative relations *entrance is adjacent to main on its west, main is wider than entrance, main is longer than entrance, main is higher than entrance.* We have restricted the height of the specific scene thus forcing the solution generator to a relatively limited number of solutions. Nevertheless, there is wide variety in the arrangement and the size of the bounding boxes. One may notice, for example that in case (iv) the width of the *entrance* is larger than the length of *main* which is valid since no declarative connection exists for these two properties.

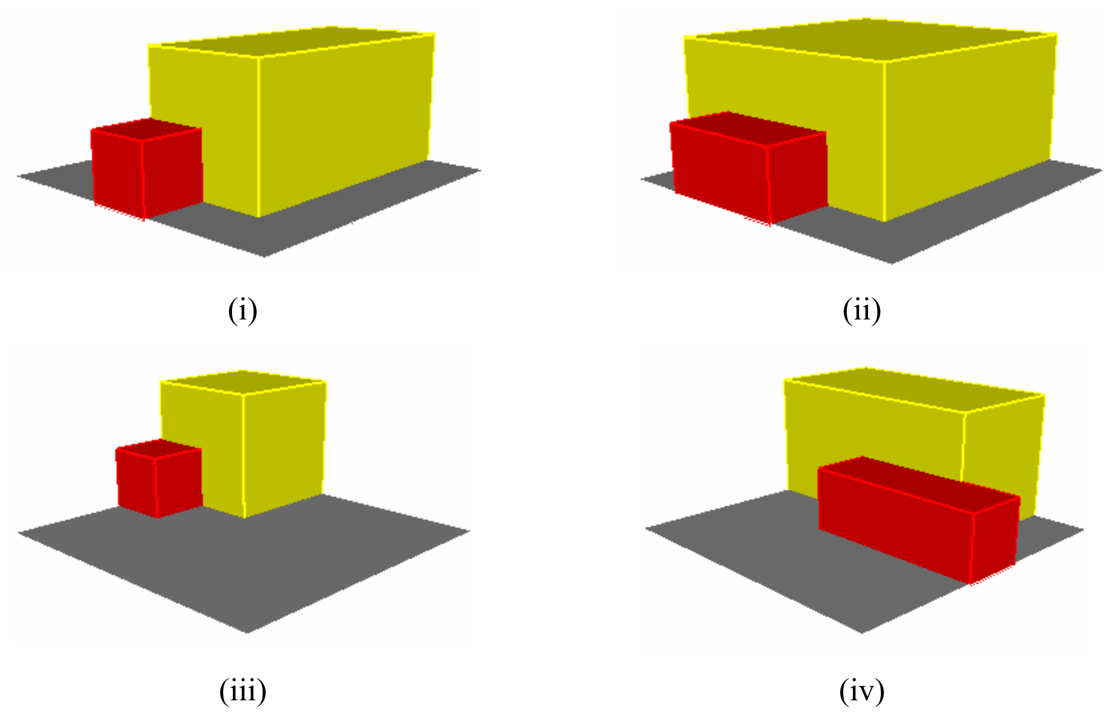


Figure 2.11 Alternative bounding box arrangements for the *igloo* habitation

Figure 2.12 presents the two stages for a single bounding box arrangement where the geometric objects used for the final representation vary. The bounding box arrangement used for the example of Figure 2.12 is (ii) of Figure 2.11.

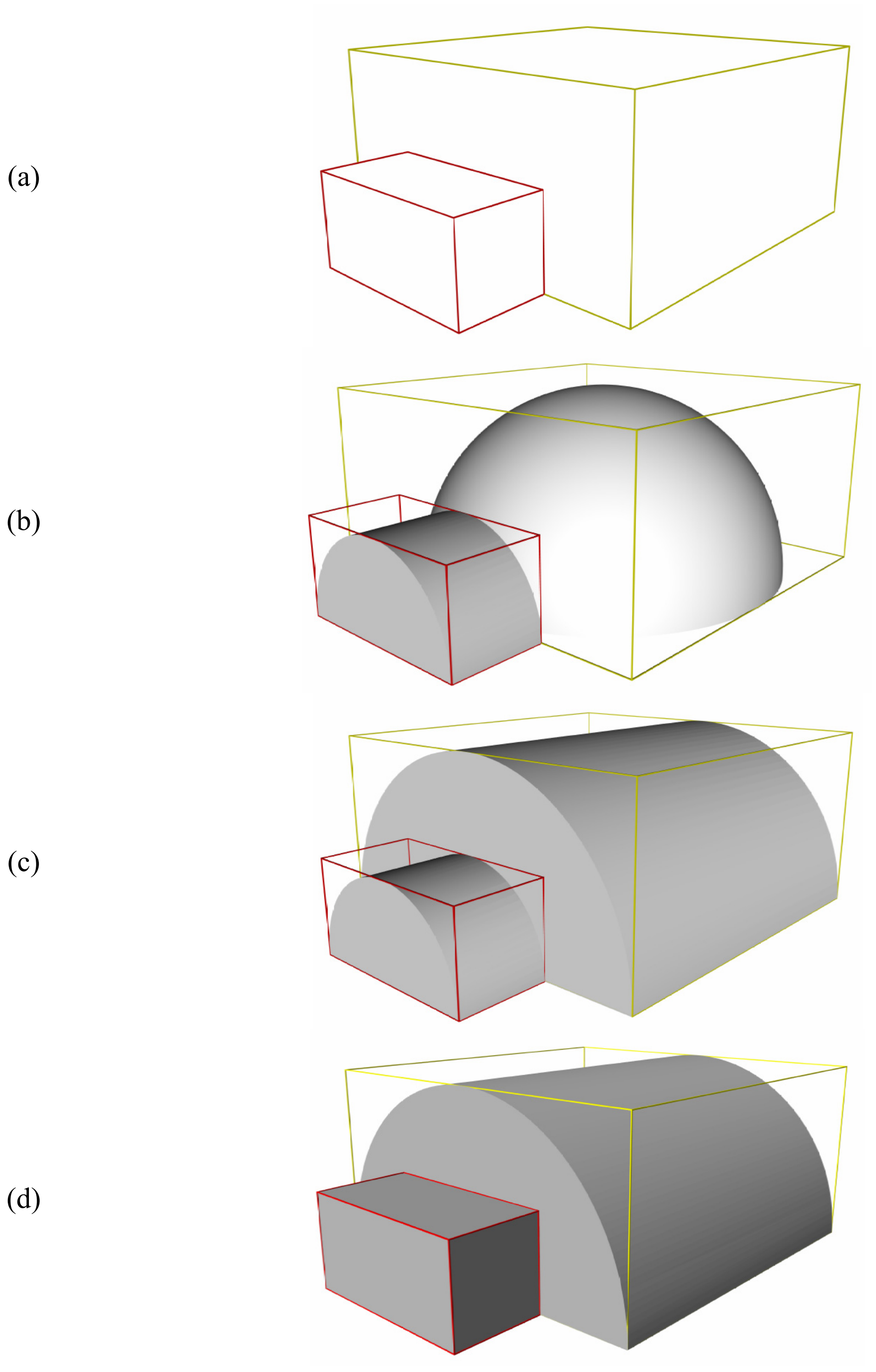


Figure 2.12 Bounding boxes arrangement (a) and final geometric representations (b),(c),(d)

2.6.5 Solution Representation

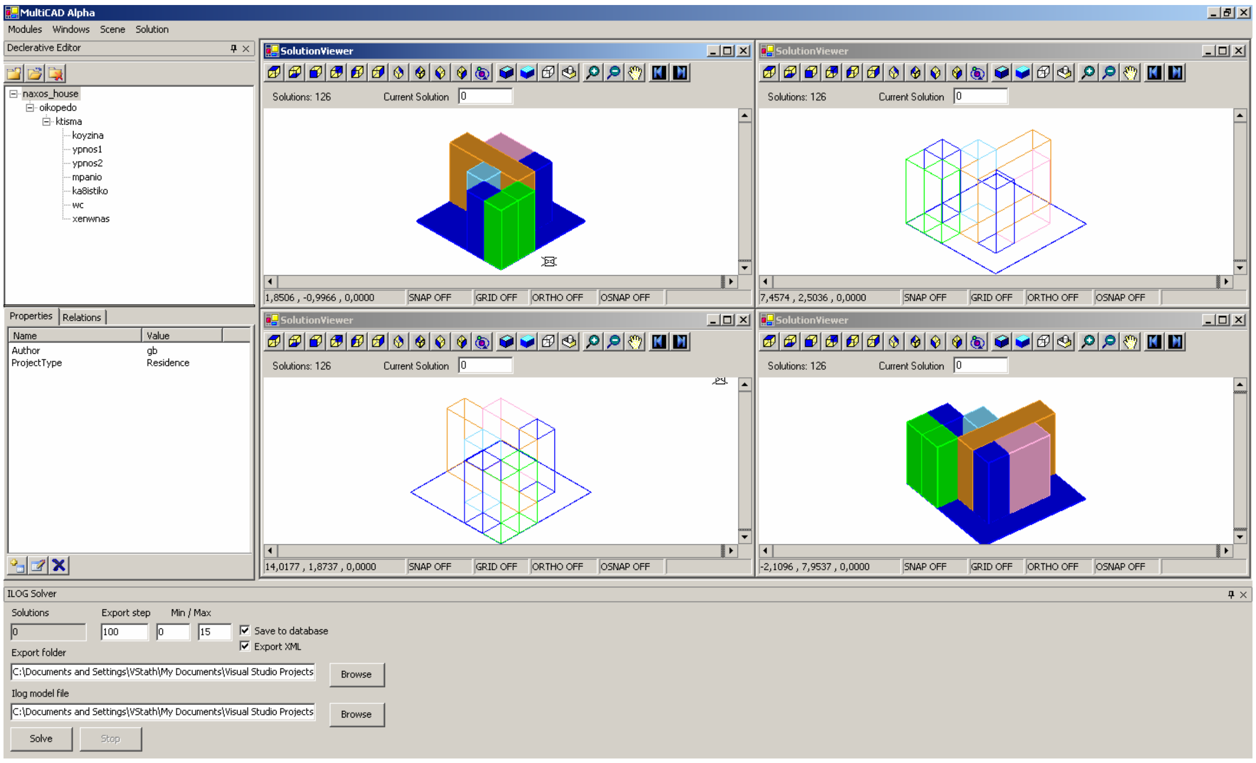
At the present stage of the Open-MultiCAD project, the geometric representation of a scene has been intentionally kept simple. In particular, each object is represented by the 3-dimensional location of its origin and the appropriate values defining its dimensions. The position of the origin with respect to the object, as well as the required dimensions may vary depending on the geometric primitive used for the representation. In order to clarify this, we have collected in Table 2.9 the properties defining an object in the geometric representation of the scene. Some additional properties, including colour, texture, material, etc. although not currently used during solution generation to produce alternative scene interpretations have been considered in the overall Open-MultiCAD architecture and already exist as potential properties in the system's database.

Geometric Primitive	*Properties*	*Comment*
parallelepiped	origin as x,y,z length width height	Origin is positioned at the south west bottom vertex of the object
hemisphere	centre as x,y,z radius	
semi-cylinder	origin as x,y,z radius direction length	Origin is the centre of the semi-circle on the west or south (depending on the direction)

Table 2.9 Example Geometric Primitives and their basic properties

2.6.6 Open-MultiCAD Interface

The current version of the Open-MultiCAD environment comprises a user interface supporting easy input of the declarative description and visualisation functionality. Figure 2.13 presents an example screenshot of the current state of the environment, exhibiting the declarative description by hierarchical decomposition on the left, a number of alternative visualised solutions on the right and the solution generator settings at the bottom.



Name	Value
Author	gb
ProjectType	Residence

Solutions: 126 Current Solution 0

1,8506 , -0,9966 , 0,0000 SNAP OFF GRID OFF ORTHO OFF OSNAP OFF

7,4574 , 2,5036 , 0,0000 SNAP OFF GRID OFF ORTHO OFF OSNAP OFF

14,0177 , 1,8737 , 0,0000 SNAP OFF GRID OFF ORTHO OFF OSNAP OFF

-2,1096 , 7,9537 , 0,0000 SNAP OFF GRID OFF ORTHO OFF OSNAP OFF

Solutions 0 Export step 100 Min / Max 0 15

[✓] Save to database

[✓] Export XML

Export folder C:\Documents and Settings\VStath\My Documents\Visual Studio Projects

Ilog model file C:\Documents and Settings\VStath\My Documents\Visual Studio Projects

Figure 2.13 The Open MultiCAD Declarative Design Environment Interface

3 Preference Modelling and Decision Analysis

The current chapter consists of two main parts: the first part presents the theoretical foundation of Preference Modelling. The corresponding notation and the underlying intuition are also discussed. The second part concentrates on Decision Support techniques, implementing the notions of preference in the context of multicriteria decision problems. In this part, the main directions in the multicriteria decision support area as well as the differences among them are presented. The mechanisms that have contributed to the module proposed and implemented in the context of the current work are also discussed in detail.

3.1 Preference Modelling

In order to extract a practical measure of the user's typically informal and immeasurable preference with respect to solutions, we present the *preference model*, using the terms of the area as presented in [Vincke92]. Our approach conforms, essentially, with other approaches in the area presented in [MCDA-SAS05] although the notation sometimes varies.

3.1.1 Preference Structure

Formally, the set of *actions* is the set of objects, decisions, candidates, etc. to be examined by the user as alternative options. In the present context, this is the set *G* of objects that comply with the declarative description submitted as input to the system. The specifics of the object generation mechanism were presented in the previous chapter and will be discussed in detail again in the next part of this thesis. For the moment, we assume that this set of objects *G* possesses two fundamental properties, namely

- it is *finite* and
- it is *stable*, i.e. not changing during the decision making process.

User *Preference, Indifference* or *Incomparability* between any two objects $s_1, s_2 \in$ G is denoted as:

$$\begin{gathered} s_1 \text{ P } s_2 \\ s_1 \text{ I } s_2 \\ s_1 \text{ J } s_2 \end{gathered}$$

respectively where P, I, J represent relations over G. Formally

$$\begin{gathered} P = \{(s_1, s_2) \mid s_1, s_2 \in G\} \\ I = \{(s_1, s_2) \mid s_1, s_2 \in G\} \\ J = \{(s_1, s_2) \mid s_1, s_2 \in G\} \end{gathered}$$

and, according to definition,

$$P \cap I = \varnothing, P \cap J = \varnothing, I \cap J = \varnothing$$

Notice that Indifference signifies equal preference whereas Incomparability signifies inability to compare for some reason; therefore they represent two distinct concepts.

In order to have a *valid preference structure* the aforementioned relations must have the following properties [Vincke92]:

$$\begin{gathered} \forall s_{1,} s_2 \in G: \\ s_1 \text{ P } s_2 \Rightarrow s_2 \not{\text{P}} s_1 \\ s_1 \text{ I } s_1 \\ s_1 \text{ I } s_2 \Rightarrow s_2 \text{ I } s_1 \\ s_1 \not{\text{J}} s_1 \\ s_1 \text{ J } s_2 \Rightarrow s_2 \text{ J } s_1 \end{gathered}$$

and, moreover

$$\forall s_1, s_2 \in G : (s_1 \text{ P } s_2) \underline{\vee} (s_2 \text{ P } s_1) \underline{\vee} (s_1 \text{ I } s_2) \underline{\vee} (s_1 \text{ J } s_2)$$

The latter implies that, for any two objects from the set of actions,

- one may be preferred to the other,
- they may be equally preferable or
- they may be incomparable.

It is worth noticing that the logical situation that arises complies with the four-valued logic [Doherty92] which has been proposed for the specific area in [Dubarle89]. Notice that the set of properties that has just been defined does not imply *transitivity* per se. In particular, it may be the case that s_1 P s_2 and s_2 P s_3 and, still, s_3 P s_1 without contradicting with the aforementioned properties. Thus, we may state that a valid preference structure does not necessarily require or imply transitivity. Nevertheless, in the following we present a number of reasons leading to the adoption of this assumption.

3.1.2 User Preference as a Function

We will use the term *user preference* for an unknown function $f{:}G\rightarrow\Re$ expressing, in direct analogy, the unique *numeric degree* of a user's preference for a specific solution. We will assume that such a function exists and we will try to approximate it through a function $p{:}G\rightarrow\Re$ by means of a number of alternative methods. This assumption is in compliance with the classical approach of *value function* or *utility function* (when cost is involved) of the Decision Analysis literature [MCDA-SAS05] and implies a number of advantages and disadvantages. Moreover, this approach complies with popular decision making mechanisms that can be applied in the current context for reasons discussed in the Decision Support chapter. These mechanisms also attach a numerical value to each solution that uniquely identifies it, encapsulating all of its interesting properties. Hence, our efforts – as far as the Decision Support Module, which is described in the second part of this thesis, is concerned – have concentrated on the construction of the function p that will approximate as much as possible the user's intuitive function f. Formally, our assumption includes the following definitions:

$$s_1 \,\mathrm{P}\, s_2 \Leftrightarrow p(s_1) > p(s_2)$$

$$s_1 \,\mathrm{I}\, s_2 \Leftrightarrow p(s_1) = p(s_2)$$

The consequences, under the light of the Preference Structure definitions presented in the previous section are the following:

1. *No two solutions are incomparable.* Formally, $J=\varnothing$. Since any two solutions are represented by two numbers they can always be compared.
2. *For any two solutions $s_1,s_2 \in G$ one of the following is true:* $(s_1 \mathrm{P}\, s_2),(s_2 \mathrm{P}\, s_1),(s_1 \mathrm{I}\, s_2)$. Since any two solutions are represented by two numbers they obey the *principle of trichotomy* of numbers.

The above ensure that we use a valid Preference Structure. Moreover, they imply that

3. *User preference is transitive, i.e.* $s_1 \mathrm{P}\, s_2 \wedge s_2 \mathrm{P}\, s_3 \Rightarrow s_1 \mathrm{P}\, s_3$. This is due to the fact that the first clause of the hypothesis implies that $p(s_1)>p(s_2)$ and the second clause implies that $p(s_2)>p(s_3)$ which, combined, imply that $p(s_1)>p(s_3)$.

The latter is a convenient consequence of the preference function since, due to the increased number of alternative options to consider, it is practically impossible to require the user to provide pair-wise comparisons for all possible pairs of solutions.

3.2 Multicriteria Decision Support

The typical context of need for a Multicriteria Decision Support methodology arises when a set of alternative options have to be evaluated against a set of diverse criteria that each contributes positively or negatively to the final outcome. The corresponding field can be considered as part of the wider area of Decision Analysis that has been established during the 1960's as the formal procedure for the analysis of decision problems [Howard66]. Several multicriteria decision support mechanisms have been proposed, including the Analytic Hierarchy Process [Saaty80] and SMART [Goodwin04] – a variation of the Multi-attribute Utility Theory [Keeny76]. A similar approach, requiring, however, minimal input on behalf of the Decision Maker, relies on standard weight assignment for the attributes [Roberts02], and is also of interest in the present context. All of these mechanisms are based on the *classical* assumption that every alternative option can be mapped to a numerical value, typically calculated as a weighted sum. An alternative direction is represented by the family of *outranking methods*, including the variations of ELECTRE [Roy68] and PROMETHEE

[Brans85] mechanisms. A thorough and up-to-date overview of the Decision Analysis field can be found in [MCDA-SAS05].

The main difference of the two aforementioned directions is their flexibility regarding the *incomparability* of two options. The weighted sum approach circumvents this notion due to its construction: any two numbers are comparable and since each alternative option is mapped to a number, all options are mutually comparable. The outranking methods, on the other hand, are flexible in regard to incomparability, a fact that suggests several benefits with respect to reality representation as discussed in the User Preference Modelling chapter but may prove to be a drawback when the number of options increases.

Both directions usually require the pair-wise comparison of alternative options, at some level, as part of their evaluation procedure. Nevertheless, outranking methods heavily rely on this information in order to provide adequate feedback. Pair-wise comparison of all alternative options is practically impossible to apply in the current context due to the prohibitive number of alternative options and the often subtle differences among them as will be discussed in the chapter covering the Intelligent User Profile Module. Hence, in the following, we have chosen to focus on the specifics of the first family of multicriteria decision support methods, covering AHP, SMART and the standard weights variation.

For the rest of the chapter, we start by introducing a basic set of assumptions and definitions. Next, we briefly present the fundamentals of outranking techniques and the families of ELECTRE and PROMETHEE methods. Subsequently, we concentrate on the weighted sum methodologies, starting by a set of definitions forming the common foundation of all three multicriteria decision support methods to be examined. We present the specifics of each approach up to the degree of detail required by the current context. In particular, we elaborate on the weight assignment policy of each method but we do not focus on other concerns as, for example, the stability analysis, since they fall outside the scope of the current work. Finally, we discuss the main differences of the methods and the effect of their adaptation to the specific context.

3.2.1 Assumptions and Definitions

What are the criteria that make one of the available solutions more preferable to another one? Focusing on the current context, the user may have approved a solution because it contains big bedrooms, rejected another because it contains a kitchen which is too narrow, etc. Moreover, the user may have also approved or rejected some solutions because (among other reasons) of the arrangement of rooms, although he/she may not be fully aware of this preference or its contribution to the selection. We can generally divide user criteria in two main categories:

- *Conscious* criteria that the user consciously employs in order to evaluate solutions.
- *Subconscious* criteria that contribute to the approval or rejection of a solution but are not realised by the user.

In the following, we concentrate on the first kind, covering the case where the criteria are well known and, therefore, can be adequately represented, whereas the second kind is discussed, up to an extent, in the chapter covering Machine Learning methods. Under the assumption of awareness of the criteria on behalf of the user, the problem of solution selection according to user preferences in the context of Open-MultiCAD can be reduced to a Multicriteria Decision problem.

In particular, a typical Multicriteria Decision problem starts with the assignment of the role of the *Decision Maker (DM)* to the person responsible for the evaluation of the alternative options and the selection of the most appropriate one(s). Next, two main components have to be determined, namely:

- The set of *actions* or *options* and
- the set of *criteria*.

The DM's objective is to select the best option(s) – or to order them in descending preference order – based on the specific criteria. In the current context, the DM is the user and the set of options is a set of solutions *G* generated based on the description submitted as input. A major step towards the solution of a Multicriteria Decision problem is, having determined the set of criteria, to map these criteria to a set of *attributes*. These attributes have to be *common for all options* and will serve as the *quantifiers of the performance* of each option against these criteria.

3.2.2 Outranking Methods

Outranking methods have been introduced in order to overcome certain difficulties that arise when attempting to apply weighted sum methodologies in specific contexts. In particular, there are cases where due to lack of information or inability of the Decision Maker to select, two or more options may be incomparable. These methods aim to capture this fact and include it in the overall evaluation of the options.

3.2.2.1 ELECTRE Methods

Similarly to the weighted sum methodologies described in the subsequent sections, in ELECTRE[1] I method each attribute participating in the options evaluation is assigned a weight. However, this weight is not used in order to calculate a weighted sum as is the case with the classical approach. The rationale is that, although there is the numerical representation of an option's performance with respect to each attribute, this number is based on different and incomparable scales for each attribute; therefore a weighted sum would not be meaningful. Instead of this, the final decision regarding the fact that *a is at least as good as b* requires that the sum of weights of the attributes where *a* outranks *b*, i.e. its grade for the specific attribute is larger than or equal to the other's, is over a certain *concordance level s* which is usually chosen to fall within the range [0.5, 1-*minimum_weight*] for normalised weights. In a sense, this sum represents the sum of the voting power (represented by their weights) of all attributes where *a* outranks *b.* An additional condition is required to hold, that of *non-discordance.* This condition requires the largest difference of grades, with respect to a single attribute, where *b* outranks *a* to be less than a certain threshold *v.* Intuitively, this condition represents the requirement that *a* should not be much worse than *b* in the context of *any* attribute.

Subsequent ELECTRE methods have tried to eliminate certain issues arising in the real world that could not be represented by the original method. In particular, ELECTRE Iv introduced the notion of *veto* as a value which, depending on each option's grade with respect to a specific attribute, would be added to its grade when calculating the discordance index. The meaning would be that, even if an option *a* outperformed *b* in many attributes, discordance with respect to a single attribute, *amplified* by a *veto function* (not an absolute number), independent of specific scales, could prevent *a* from being announced as *a is at least as good as b.* ELECTRE IS further enhanced the concordance and non-discordance conditions with pseudo-criteria in order to exploit fine differences of preference thus resolving indifference between options and yielding, as a result, a minimal set of selected options.

[1] Acronym from the French title: ELimination Et Choix Traduisant la REalité

An alternative direction of the ELECTRE methods has aimed to the complete ordering of the available options from the most to the least preferable. ELECTRE II was the first of these methods, largely based on ELECTRE Iv. The main difference from the latter is that there were now two distinct concordance levels, representing *strong* and *weak* outranking, which are used repeatedly to partition the set of options to classes of preference. ELECTRE III was introduced as an improved version of ELECTRE II, able to handle inaccurate or ill-defined data through the use of pseudo-criteria and a *credibility index* assigned to each outranking decision between any two options. ELECTRE IV applied a set of five outranking relations each representing a different degree of credibility in order to overcome the requirement for attribute weight assignment on behalf of the Decision Maker. Finally, ELECTRE TRI was introduced as a classification method, aiming to assign each option to a specific class, without relative consideration of alternative options. Instead of that, representative options for the *upper* and *lower* limit of every category were used and each option was evaluated against them according to the ELECTRE III methodology.

3.2.2.2 PROMETHEE Methods

The family of PROMETHEE methods is largely based on the similar to ELECTRE methods' assumption that it is unacceptable to combine the performance of an option in the context of two or more attributes, usually measured in different scales, in order to obtain the option's overall evaluation. Therefore, the additional information required by the method concerns the comparison of the attributes themselves and the varying degrees of preference within the context of a single attribute at a time. The idea of PROMETHEE II method is that an option *a* should be considered preferable to an option *b* when the algebraic sum of preference difference for all criteria and all other options is larger for *a* than for *b.* Intuitively, this algebraic sum represents the algebraic sum of the intensity of *a* when outranking other options in every attribute minus the intensity of *a* when being outranked by other options in every attribute. This algebraic sum actually yields a numerical value that is used to produce a complete order of all alternative options. When considered separately, the two parts of the sum, i.e. the *outrank others* and *be outranked by others,* lead to the partial ordering used by the PROMETHEE I method. The PROMETHEE V variation comprises an alternative methodology, applied whenever the requirement is to select a subset of the options without any particular interest in their order.

3.2.3 Attribute Tree

All weighted sum methodologies examined in the subsequent sections rely on a set of attributes, appropriately organised in a hierarchical structure. In particular, attributes are defined by gradually constructing an *attribute tree* in a top-down manner. In our context the procedure starts with the root of the tree – Level 0 – which is the main goal of maximising user's preference. Notice that user's preference represents, in the current context, an option's performance according to the DM. The general criteria that define this preference are translated to the initial general attributes yielding Level 1. These, in turn, are analysed to more specific attributes. This process is repeated, if necessary, until the lowest level attributes can be directly derived by the explicit properties of each option.

The attribute tree should fulfil the following requirements [Keeny76]:

- **Completeness.** The tree includes all attributes of interest to the DM.
- **Operationality.** The DM can easily evaluate the lowest level attributes for each solution.
- **Decomposability.** The evaluation of a solution with respect to an attribute should not depend or influence its evaluation against another attribute.
- **Absence of Redundancy.** No two attributes represent the same feature.
- **Minimum Size.** The tree has the minimum size possible, not affecting fulfilment of the previous requirements.

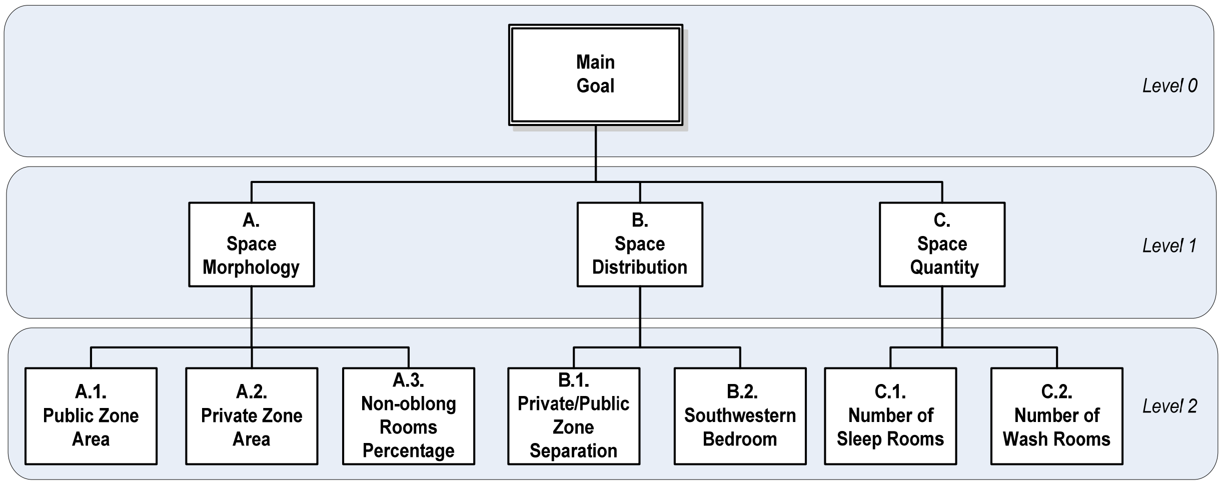

Main Goal
Level 0
A. Space Morphology
B. Space Distribution
C. Space Quantity
Level 1
A.1. Public Zone Area
A.2. Private Zone Area
A.3. Non-oblong Rooms Percentage
B.1. Private/Public Zone Separation
B.2. Southwestern Bedroom
C.1. Number of Sleep Rooms
C.2. Number of Wash Rooms
Level 2

Figure 3.1 Example Attribute Tree

Figure 3.1 is an example attribute tree as described above, based on selected criteria and resulting to the set of corresponding attributes. The specific attributes are discussed in detail in a next chapter, covering the proposed and implemented Intelligent User Profile Module. It is worth noticing that according to the definition of *completeness,* the same set of options may be evaluated by a different attribute tree depending on the DM's criteria. In other words, the kind of options to be evaluated, e.g. building assemblies in the current context, does not necessarily suggest a unique set of attributes to be considered. As long as the other properties are preserved, completeness is at the discretion of the DM.

3.2.4 Analytic Hierarchy Process – AHP

The specific method has been initially described in the early seventies and has drawn considerable attention, being applied to diverse economic, governmental and commercial areas [Saaty80], implemented in commercial products [ExpertChoice05] and triggering interesting variations as in [Buckley85] and ANP [MCDA-SAS05]. In this section we describe the basic steps of AHP focusing on the points of interest within the current context.

The AHP is based on four axioms:

- Reciprocal judgments.
- Homogeneous elements.
- Hierarchic or feedback dependent structure.
- Rank order expectations.

Axiom (3) implies conformance to the aforementioned attribute tree, which AHP actually extends with an additional level as described below. Compliance with the rest of the axioms is mentioned within the respective context in the discussion that follows.

In particular, after the construction of the attribute tree the method requires the extension of each attribute with the hierarchy of the options *according to the specific attribute,* thus adding an extra level under each attribute, containing all available options. The children of every node of the tree are supposed to be ordered according to their *importance* – or *preference* in the case of the ordering of options – for the DM. This implies that, regarding the general (A,B,C) or specific (A.1, A.2, etc.) attributes of the example of Figure 3.1, four different orderings are required, namely:

- Ordering of general attributes A, B, C,
- ordering of specific attributes A.1, A.2, A.3,
- ordering of specific attributes B.1, B.2 and
- ordering of specific attributes C.1, C.2.

Moreover, seven more orderings are required, since all available options have to be ordered with respect to each specific attribute separately.

All of these orderings are achieved by means of a pair-wise comparison mechanism which is described in detail below. In the following we will refer to the *items* compared, implying the options or the attributes, since the mechanism is exactly the same for both cases.

3.2.4.1 Pair-wise Comparison Matrix

In order to provide an overall ordering of the n items of a specific level, a square matrix M is constructed containing one row and one column for each item (option or attribute) to be evaluated. The order of the items is the same for the rows and columns of M. Each cell of the matrix must be filled with a number $r_{i,j}$ revealing the *ratio of preference (or importance)* between the item corresponding to the specific row i and the item corresponding to the specific column j. According to the method, the acceptable values are

$$r_{i,j} \in \{\frac{1}{9},\frac{1}{8},\frac{1}{7},\frac{1}{6},\frac{1}{5},\frac{1}{4},\frac{1}{3},\frac{1}{2},1,2,3,4,5,6,7,8,9\}, \forall i,j \in \{1,\ldots,n\}$$

The original values included only odd numbers (and the corresponding fractions) representing verbal ratios of importance as follows:

- equally important: 1
- weakly more important: 3
- strongly more important: 5
- very strongly more important: 7
- extremely more important: 9

Nevertheless, intermediate values may also be used to reveal slight variations of the degree of preference or importance. It easily becomes apparent that the diagonal of the matrix represents the ratios for each item against itself hence it should contain 1's. Moreover, symmetric cells, with respect to the diagonal, should contain reciprocal numbers since they refer to the same pair of items in reverse order. Formally, this is an implication of the aforementioned Reciprocal Judgements axiom (1) assumed by the method. Moreover, the ability to compare different items (attributes or options) according to a uniform scale of ratios of importance implies compliance with the Homogeneous Elements axiom (2). Therefore, the DM has to complete only the lower (or upper) part of the table since each half – below or above the diagonal – is implied by the other. Table 3.1 shows an example of the completed pair-wise comparison matrix regarding the importance of attributes under general attribute Space Morphology appearing in Figure 3.1.

	A.1 ***Public Zone Area***	***A.2*** ***Private Zone Area***	***A.3*** ***Non-oblong Rooms Percentage***
A.1. Public Zone Area	1	3	9
A.2. Private Zone Area	$\frac{1}{3}$	1	5
A.3 Non-oblong Rooms Percentage	$\frac{1}{9}$	$\frac{1}{5}$	1

Table 3.1 Example Pair-wise Comparison Matrix for the Space Morphology attributes

Due to the pairing of attributes, this method may lead to a certain type of inconsistency regarding the product of importance ratios. In particular, considering the example of Table 3.1, since the Public Zone Area (A.1) has been rated as 3 times more important than the Private Zone Area (A.2) and the latter has been rated, in turn, 5 times more important than Non-oblong Rooms Percentage (A.3), one might expect that the Public Zone Area (A.1) should – automatically – be considered 3×5=15 times more important than Non-oblong Rooms Percentage. Nevertheless, the DM is free to assign another ratio for A.1 against A.3 and not the implied 15. It is worth noticing that this fact has actually caused some criticism [Goodwin98] to the formal AHP approach since 15 is not one of the available ratios.

3.2.4.2 Normalised Weights

After the pair-wise comparison table *M* has been completed, the set of weights can be calculated for the corresponding items. The weight extraction method is based on the calculation of the principal right eigenvector of *M* and the corresponding principal eigenvalue which, due to the construction of the table, is the only non-zero eigenvalue. Practically, both are calculated based on the fact that the sequence M^k converges to a matrix with identical columns, proportional to the principal right eigenvector of *M* as *k* grows. Setting $k>10$ is typically enough for adequate precision, therefore the algorithm for weight extraction consists of the following steps:

1. Calculate M^k, for a k>10.
2. Let *C* be any of the *n* columns of the square matrix M^k. Calculate weights as:

$$w_i = \frac{c_i}{\sum_{r=1}^{n} c_r}$$

i.e. the weights are the normalised elements of any column of the table after the latter has been raised to an adequately large power. For example, applying the above calculations to the matrix *M* obtained from Table 3.1 yields (w_1, w_2, w_3) = (0.672, 0.265, 0.163) for k=16.

3.2.4.3 Inconsistency Ratio

The next step of the AHP method is the calculation of *Inconsistency Ratio* (also appearing as *Consistency Ratio* in the literature) that reveals to what extent the user has not respected transitivity during the completion of the pair-wise comparison table. This ratio is based on the eigenvalues of the positive reciprocal matrix given as input. In particular, if the user were perfectly consistent during the completion of the table, the maximum eigenvalue of the resulting matrix should be equal to its dimension whereas the non-principal eigenvalues should all be equal to 0. Formally, for the maximum eigenvalue $\lambda_{\max}$ the following equality is true:

$$M \cdot w^T = \lambda_{\max} \cdot w^T$$

where w^T is the transposed weight vector. The formula above implies that only one row of *M* is needed for the calculation of λ_{max} and the subsequent Inconsistency Ratio *IR* which is, formally, the average non-principal eigenvalue of *M*. Using the first row of the matrix for the calculations the next steps of the algorithm are:

$$\lambda_{\max} = \frac{\sum_{i=1}^{n} m_{1,i} w_i}{w_1}$$

$$IR = \frac{\lambda_{\max} - n}{n-1}$$

where n is the dimension of the matrix. For the example input matrix of Table 3.1 we have λ_{max}=3.029 and IR=0.015 whereas, ideally, they should be 3 and 0 respectively.

3.2.4.4 Combining Results

After the pair-wise comparisons for all levels have been completed the overall grades for all options can be calculated. Notice that according to the method, the pair-wise comparison is extended to the options themselves, in the context of each attribute. Therefore, after this stage the performance of each option against each attribute is available as a normalised value. Thus, each option may receive an overall grade as a weighted sum of the grades of the specific option, using the attribute weights. Formally, for an option s having received the following grades for each lower level attribute,

$$g_s=(g_1,g_2,\dots,g_n)$$

the corresponding overall evaluation is

$$b = w \cdot g = \sum_{i=1}^{n} w_i g_i$$

where n is the total number of attributes.

3.2.5 Simple Multi-Attribute Rating Technique – SMART

After the construction of the attribute tree, SMART method [Goodwin04] applies a simple *swing weights* technique for the assignment of weights to each individual lower level attribute. In particular, the DM is first required to answer the following question:

"Assume an option that exhibits the worst performance for every attribute. If you had the ability to maximise its performance for only one attribute, which attribute would you choose?"

Based on the answer to this question the *most important* attribute a_{top} is selected. Then, in order to comparatively order the remaining attributes, the following question is repeated for each remaining attribute a_i:

"Assume an option that exhibits the worst performance for every attribute. What percentage of improvement does maximisation of performance for attribute a_i represent when compared to the 100% improvement represented by the maximisation of performance for a_{top}?".

Thus, all attributes are mapped to a number ranging from 0 to 100 as shown in the corresponding column of Table 3.2. Weight normalisation yields the final attribute weights appearing in the next column of the same table.

Attribute	***Swing Weight***	***Normalised Weight***
Public Zone Area	100	0.29
Private Zone Area	90	0.26
Non-oblong Room Percentage	20	0.06
Private/Public Zone Separation	30	0.09
South-Western Bedroom	10	0.03
Number of Sleep Rooms	60	0.18
Number of Wash Rooms	30	0.09
Total	*340*	*1.00*

Table 3.2 Swing and normalised attribute weights as obtained according to SMART

As soon as the normalised weights are available, the overall performance of each option is calculated as a weighted sum. In particular, the SMART method makes use of the middle values, actually extending them to *quarter points¸* representing 25% and 75% performance for a specific attribute. Based on these points, a *value function* is defined using a simple extrapolation scheme in order to allow evaluation of all intermediate values. This value function combined with the normalised weights leads to a unique *grade* for each option calculated as a weighted sum.

3.3 Standard Weights Assignment

Standardised weight assignments have also been proposed to eliminate the need for explicit assignment by the DM [Roberts02]. These methods operate under the assumption that only the *order* of importance suggested by the DM is of significance and not the *degree* of importance per se. According to this approach, weights are calculated and assigned based on simple mathematical functions that depend only on the ranking of an attribute like the Rank Order Centroid (ROC), Rank Sum (RS), Rank Reciprocal (RR) and Rank Order Distribution (ROD). [Roberts02]. In particular, these methods require only the importance ranking R_i of the attributes by the DM and, based on their total number n, they automatically assign weights according to the formulas shown below, assuming $R_i=1$ for the most important attribute and $R_i=n$ for the least important.

- Rank Order Centroid: $$w_i = \sum_{k=R_i}^{n} \frac{1}{k}$$
- Rank Sum: $$w_i = \frac{2(n - R_i + 1)}{n(n+1)}$$
- Rank Reciprocal: $$w_i = \frac{1}{R_i \cdot \sum_{k=1}^{n} \frac{1}{k}}$$

The ROD method proposed in [Roberts02] requires more complex calculations, however, the comparative analysis presented therein shows ROD and RR as the most efficient of these weight approximation methods. Table 3.3 presents example weight values for a range of attribute populations according to the Rank Reciprocal standard weight assignment function. Notice that the weights produced are already normalised due to the function construction.

	Total Number of Attributes						
Rank	***2***	***3***	***4***	***5***	***6***	***7***	***8***
1st	0.667	0.545	0.480	0.438	0.408	0.386	0.368
2nd	0.333	0.273	0.240	0.219	0.204	0.193	0.184
3rd		0.182	0.160	0.146	0.136	0.129	0.123
4th			0.120	0.109	0.102	0.096	0.092
5th				0.088	0.082	0.077	0.074
6th					0.068	0.064	0.061
7th						0.055	0.053
8th							0.046

Table 3.3 Standard Weights according to Rank Reciprocal function for 2 up to 8 attributes

3.4 Concluding Remarks

User evaluation of geometric object representations in the context of the Open-MultiCAD environment has triggered the examination of User Modelling fundamentals and Decision Support mechanisms that could contribute towards this direction. Two main axes in this area exist: the *classical approach*, where user preference is assumed to be represented by a weighted sum that maps each option to a unique numerical value and the *outranking approach*, where options are evaluated based on mutual comparison. Representatives of the former approach include the AHP and SMART methods whereas the main representatives of the latter are the ELECTRE and PROMETHÉE methods.

The increased number of solutions produced for a typical declarative description in the context of the Open-MultiCAD environment prohibits the mutual comparison of the alternative solutions which is the essence of outranking methods. Therefore, we have decided to apply techniques based on the classical approach in order to automate, up to an extent, solution evaluation with minimal user intervention. In particular, we have chosen to incorporate in the proposed and implemented Intelligent User Profile Module, a mechanism for the assignment of weights to a set of observed attributes for the solutions to be evaluated. This mechanism is based on the weight assignment techniques applied in AHP and SMART methods and also employs the Rank Reciprocal function for this purpose. These three weight assignment techniques demand user feedback that varies from detailed (AHP) to average (SMART) and very simple (RR) thus covering the whole range of required user intervention for the specific task.

4 Machine Learning

In this chapter, we start with a general discussion regarding the inherent difficulties of acquisition and application of user preferences in the current context. Next, we present the fundamentals of Intelligent Information Systems. We continue with the presentation of the most important Machine Learning mechanisms available of relevance to this work. Finally, we explain the rationale supporting the choice of the particular family of algorithms that has formed the basis of the Machine Learning Component proposed and implemented as part of the Intelligent User Profile Module.

4.1 Introduction

Human behaviour can be confusing, contradicting or even irrational in several occasions and for diverse reasons. Although universal notions like the survival instinct or love and their variations are usually considered as the controlling forces of our behaviour, they often fail to explain the way we react to important as well as trivial matters. Either because human entities and their environment are too complicated to be simulated and measured respectively, or because free will can not be accurately predicted (otherwise it should not really be considered free) it seems that, at least considering the present status of human knowledge and the corresponding technical capacity, a deterministic non-human system is not able to simulate human behaviour in its entire range. The latter becomes also evident by the fact that Turing's Imitation Game, has yet to be successfully passed by any artificial mechanism unlike its inventor's predictions [Turing50].

It follows that, if we still need to simulate human behaviour for experimental or other reasons, we have to accept the idea of a compromise represented by reduced model accuracy and/or a set of approximative assumptions. The impact of these observations is twofold. First, it is implied that, in experimental conditions where human feedback is not readily available due to limitations on time, effort, cost, etc., any artificially generated substitute feedback will lack accuracy. Second, even if a finite amount of human feedback is available and has been optimally exploited by a learning mechanism, there is no guarantee that the resulting model will successfully simulate future feedback originating from the same human source.

The effort to capture and apply user morphological preferences comprises part of the effort to simulate and predict human behaviour. In particular, the ability of a system to learn from experience forms an integral part of its intelligent behaviour. However, unlike cases where an underlying function, unknown but existing, is to be learnt based on existing data, capturing user preferences presents an even higher degree of difficulty. We outline, in the following, the main machine learning mechanisms and next we focus on the algorithms we have applied and adapted in order to achieve intelligent solution selection in the context of the Open-MultiCAD Design Environment. Finally, we determine the degree of accuracy we demand from our model and the assumptions we make while pursuing it.

4.2 Intelligent Information Systems

Several approaches have been applied, since the dawn of the Artificial Intelligence field in the early 1950's, aiming to endow artificial systems with intelligent behaviour. The initial ambitious visions of building robots that would act and react like humans to any external stimulus have gradually been replaced by efforts to imitate human behaviour in selected fields – image recognition, natural language processing, etc. – with carefully set objectives and respectively realistic expectations. This transition has been characterised as the shift from the *romantic* to the *modern* period in the history of AI [Jackson86],[McCarthy95]. It has now become apparent that human beings and their functions are far more complex than what we used to believe some decades ago. In another sense, it has also become apparent that current technical and algorithmic capabilities are not adequate[2] for the specific task in its entirety.

[2] Whether they may ever be seems to cross the boundary between Artificial Intelligence and Philosophy.

There are different directions that may be followed for the construction of an intelligent system, depending on the nature of the problem and the quality and quantity of the available data. In particular, there are two families of mechanisms regarding systems exhibiting intelligent behaviour. These families are often combined to produce hybrid approaches, supported by the corresponding machine learning mechanisms, hence they somehow represent the two extremes of the range [Kechman01].

White Box Approach. A set of rules are embedded in the system based on already existing knowledge about the concept. These rules may be applied to provide intelligent behaviour to the system and they are understandable by humans. Typically, the rules remain static throughout the use of the system or they are manually updated by humans. The performance of a mechanism of this type depends on the quality of the rules it comprises.

Black Box Approach. An intelligent mechanism is constructed based on the available example data. The mechanism is able to respond in an intelligent way but the knowledge it acquires and uses for this behaviour is in a form not directly apprehensible by humans. Typically, after a pre-defined *training period,* this mechanism also remains static. The performance of the mechanism depends on the quality and quantity of the data used during the training period.

The aforementioned approaches represent the two extremes of the wide range of intelligent systems, the emphasis being on knowledge representation rather than the learning process itself. In fact, both of the above definitions represent extreme cases where an intelligent system is built and subsequently does not exhibit machine learning capabilities at all in order to enhance its intelligence. However, as it is most often the case, the aforementioned approaches are extended towards this direction. An example of this enhancement is *automatic rules inference* where new rules are generated based on newly acquired examples regarding the concept under examination. Moreover, the two approaches may be combined to produce hybrid systems. An example of such a system is the case where a set of rules is automatically generated by a rule inference module whereas a separate module is responsible for assigning and maintaining a weight for each rule according to current and future examples. In other words, an intelligent system is invariably connected with learning capabilities.

Integrated intelligent systems rely on four equally important, yet distinct, tasks even though there are cases where some of these tasks are omitted. In particular,

- Real world information is translated to data that can be stored and processed by a computer system. This stage is connected with sensor technology, sampling techniques, analogue-to-digital conversion as well as input explicitly formulated by humans.
- Already existing knowledge about the concept to be learnt is encoded as a set of rules by human experts.
- Already existing as well as new data, and the *experiences* or *facts* they represent, are exploited in order to improve the intelligent system's behaviour. This stage is connected with machine learning algorithms and methodologies.
- The system takes advantage of the results of the aforementioned tasks in order to exhibit intelligent behaviour.

These tasks are typically supported by the appropriate data repository and user interface module. Figure 4.1 summarises the operation of a typical Intelligent Information System.

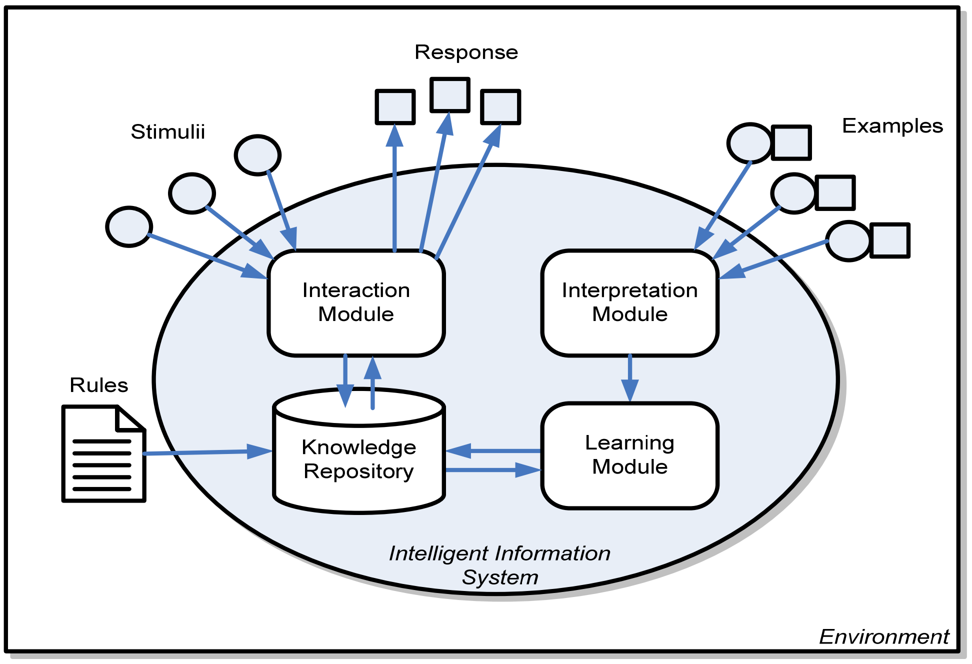


Figure 4.1 Typical Components and Interaction with the Environment of an Intelligent Information System

Practically, due to the nature of the majority of the problems posed in the current context, the information that has to be processed is machine generated and/or it originates from direct user input to a computer system environment. In these cases, since data are readily available in exploitable electronic form, the first of the aforementioned stages is practically omitted. Moreover, in cases where prior knowledge about the concept does not exist, the second stage may also be omitted. An example of such a problem, where the first two tasks are practically redundant, is that of learning web browsing user preferences based exclusively on actual user navigation. In that case, indeed, information regarding user preferences consists solely of URL's of visited web pages and their content, sequence of browsing, time spent at each one of them, time of the day and possibly some additional parameters. All this information is already available in electronic form and, regarding a transparent system, no additional information on behalf of the user is required. Similarly, in the current context, the task of capturing and utilising user preferences to improve the response of the Open-MultiCAD environment belongs to an analogous category of learning problems. For this reason, in the following, we concentrate on the third of the aforementioned tasks, discussing some of the most efficient algorithms for machine learning and their applicability to the current context.

4.3 Machine Learning in Intelligent Information Systems

Machine Learning stands as one of the dominant subjects of modern AI and a core characteristic of intelligent systems. Although the interpretation of the term varies in literature, e.g. [Konnar00], [Kechman01], [Baldi01], we have reached to the conclusion that, in general, it represents *a set of techniques for knowledge acquisition, elicitation and subsequent application of this knowledge in the context of a partially or entirely unknown concept.*

Several categorisations have been established regarding the nature of the machine learning problems [Burbidge01]. Most of them reflect differences with respect to the set of *examples* or *samples* used for the learning process. In particular, there are cases where a set of examples is available without any further characterisation regarding the unknown concept. In these cases we deal with a problem of *unsupervised learning* where we expect the employed mechanism to recognise similar samples and group them without prior knowledge regarding the number or the qualitative differences of these groups. In other cases, each example is accompanied by a characterisation revealing the class it belongs. These are cases of *supervised learning* in the sense that the mechanism is provided with information regarding the number of the different classes and the nature of the members they contain. The specific example represents a problem of *classification,* where each current or future sample should be assigned to one of a finite number of discrete classes. An alternative situation, where each sample is mapped to a real numbered value is that of *regression.* In fact, both categories of supervised learning problems may be reduced to that of approximating a multivariate function [Kechman01].

A crucial assumption in the context of supervised learning is that the attributes used for the description of the samples, i.e. the *input*, are actually *related* with the class or value corresponding to the sample, i.e. the *expected output*. This task of selecting or constructing the appropriate attributes to represent the given examples, i.e. *feature selection* or *feature extraction,* is far from trivial and is usually aided by already existing domain knowledge for the concept to be learnt and/or dedicated mechanisms. In the following we will assume that such a relation between the input and the expected output exists for at least *some* of the attributes used for the description. We elaborate on this in a later section discussing the selected approach for the current context. Moreover, since the current context refers to the approval or rejection of generated solutions, or, in a wider interpretation, the characterisation of each solution regarding user preferences, we concentrate on the most important mechanisms available for *supervised learning* for *classification.*

4.3.1 Definitions

Before we examine the specifics of the most important machine learning mechanisms it is useful to clarify a few terms used in the following discussion. For the rest of this chapter it will be assumed that the mechanisms discussed accept an *input* – usually represented by a vector of values, one for each *attribute* describing the concept to be learnt – and produce an *output* – represented, in general, by another vector of one or more values. Moreover, the mechanisms described below – or their version of interest for the current context – rely on *examples* (also referred to as *samples*) in order to establish and improve their intelligent behaviour. By the term *example* we refer to a correct (reflecting the real world concept or, at least, the desired by the designer of the mechanism) pairing of an *input* and a *target output* vector. In other words, an *example* represents the *output* the mechanism *should* produce when triggered with the corresponding *input*. Usually a set of examples is used during the *training* of the mechanism and an alternative set of examples is used for the *evaluation* or *testing* of the mechanism. The idea is that the mechanism is expected to perform well on already seen examples but it also has to demonstrate adequate performance on yet *unseen* examples. The performance of a mechanism with respect to previously unseen examples exhibits the ability of the mechanism to *generalise* the knowledge it has acquired during its training. There are cases where a mechanism fails to adequately generalise because, during training, it has modelled too closely the specific examples included in the training set. This is a case of *overfitting* the contents of the training set. On the other hand, increased flexibility of a mechanism with a view to adequate generalisation ability may lead to *underfitting,* a case where the mechanism fails to perform adequately even for the contents of the training set itself. These two cases represent the extremes of a traditional trade-off of machine learning called the *bias-variance dilemma.* The *performance* of a mechanism is assessed based on a selected set of examples and is usually expressed as a *success rate*, i.e. the percentage of examples *correctly* evaluated by the mechanism, or an *error rate,* i.e. the percentage of examples *incorrectly* evaluated by the mechanism. Mechanisms of reduced complexity demonstrate reduced ability of learning the available examples thus demonstrating *high bias (*or *high empirical risk)* and, since they are rather insensitive with respect to the training data they may generalise adequately, thus demonstrating *low variance (*or *low confidence interval)*. On the contrary, mechanisms of increased complexity are able to closely model the available examples thus demonstrating *low bias* but they often overfit the training examples thus yielding *high variance* error, i.e. low generalisation ability.

4.3.2 Artificial Neural Networks

According to a phrase ascribed to J.S.Denker and appearing often in the relevant literature *"neural networks are the second best way to do almost anything"*. The sentence implies that the (first) best way is to assign to one or more humans the tasks of understanding the different parameters of the problem, study its nature and implement a solution or a model closely resembling the reality; unfortunately, these tasks usually require a restrictive amount of time. In other words, neural networks have been considered the best way to exploit previous experience, i.e. existing examples of a concept, without necessarily delving into the nature of the concept itself in the form of rules or human understandable modelling.

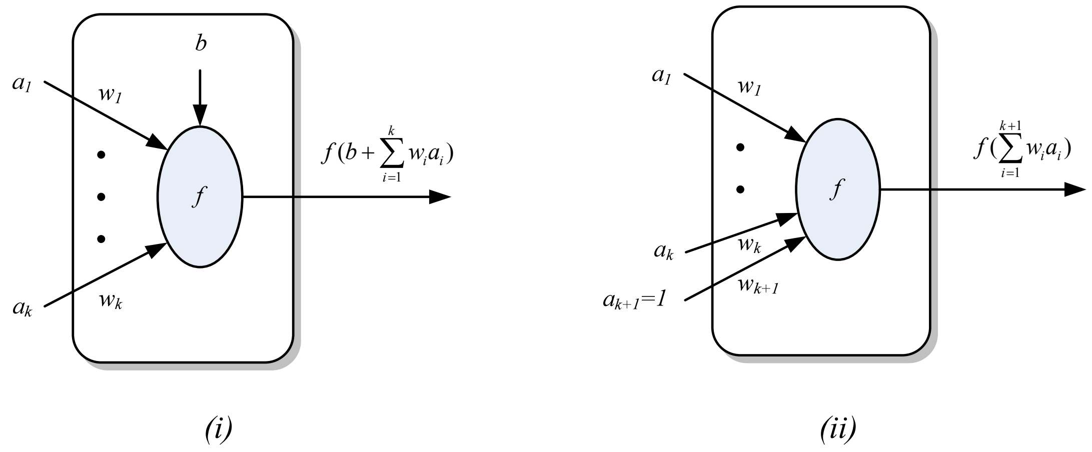


Figure 4.2 Typical artificial neuron of *k* inputs with bias *(i)* explicitly shown as *b* or *(ii)* represented by the weight w_{k+1} of an additional input of fixed value

Artificial Neural Networks (ANN), or simply *Neural Networks* (NN), since the context usually makes their artificial nature obvious, have been inspired by the structure and operation of the human brain. Their aim is, through a simplified model of the architecture and operation of the latter, to imitate, at least up to a reasonable extent, the human ability to learn from examples, memorise concepts and exhibit intelligent behaviour with respect to previously unknown concepts. Figure 4.2 shows the basic configuration of a typical artificial *neuron*, which is the building block of artificial neural networks. An artificial neural network is practically a weighted directed graph where each node maps its *incoming* edge(s) to its *outgoing* edge(s) in a specific manner. Most often, the nodes of this graph are organised in *layers,* the outputs of the neurons of one layer being submitted as input *only* to neurons of the next layer. External inputs are received by a special layer of neurons, the *input layer,* whereas the final outputs are produced by the neurons of the *output layer.* Additional *hidden* layers may exist between the input and the output layers. Due to the nature of information transmission in such a neural network, i.e. gradually, from the first layer to the last, a network of this architecture is characterised as a *feed-forward* network [Hornik89], [Leshno93]. Usually the graph is acyclic, however, when this is not the case, the network is additionally characterised as *recurrent.*

The neurons of the input layer have a neutral behaviour, simply transferring the values of the input vector to one or more neurons of the next layer. Each neuron of the other layers, i.e. *hidden* and *output*, features a number of inputs a_i each accompanied by a weight w_i and a single additional fixed input, the *bias b*. Input values may be discrete binary, may range between 0 and 1 or may range between -1 and 1, without any of these being an obligatory requirement. Weight value and biases usually range between -1 and 1. Due to the layered architecture, the values of each neuron's inputs a_i are obtained from the outputs of the neurons of the preceding layer – except for neurons of the input layer which receive their input directly from the input vector. The weights and biases, on the other hand, are only modified during training, according to specific rules, and they are considered as part of the configuration of the network which is finalised and fixed as soon as the training is over. Intuitively, each neuron merges its inputs, their weights and its bias to a single value represented by its calculated output. This calculation is suggested by the *transfer* (or *activation*) *function f* which is, in general, common for all neurons of the network or, at least, common for all neurons of the same layer. Several transfer functions *f* exist, exploiting in different manners the weighted sum *S* of the neuron's inputs and the bias, where

$$S = b + \sum_{i=1}^{k} w_i a_i$$

For example:

- Step Function: $f(S) = \begin{cases} 1, S \geq t \\ \\ 0, S < t \end{cases}$
- Sign Function: $f(S) = \begin{cases} 1, S \geq t \\ \\ -1, S < t \end{cases}$

where *t* is a fixed threshold value, not modified during training. An alternative commonly used function is also

- Sigmoid Function: $f(S) = \dfrac{1}{1+e^{-c \cdot S}} \in (0,1)$ or $f(S) = \dfrac{2}{1+e^{-c \cdot S}} - 1 \in (-1,1)$

where *c* is a constant positive coefficient adjusting the *steepness* of the sigmoid function. Figure 4.3 shows a typical neural network configuration comprising an input layer, one hidden layer and an output layer. This network is *fully connected,* i.e. the output of each neuron is submitted as input to all neurons of the next layer, as is usually the case. The specific example could represent a classifier, mapping each sample represented by three attributes to one of two alternative classes.

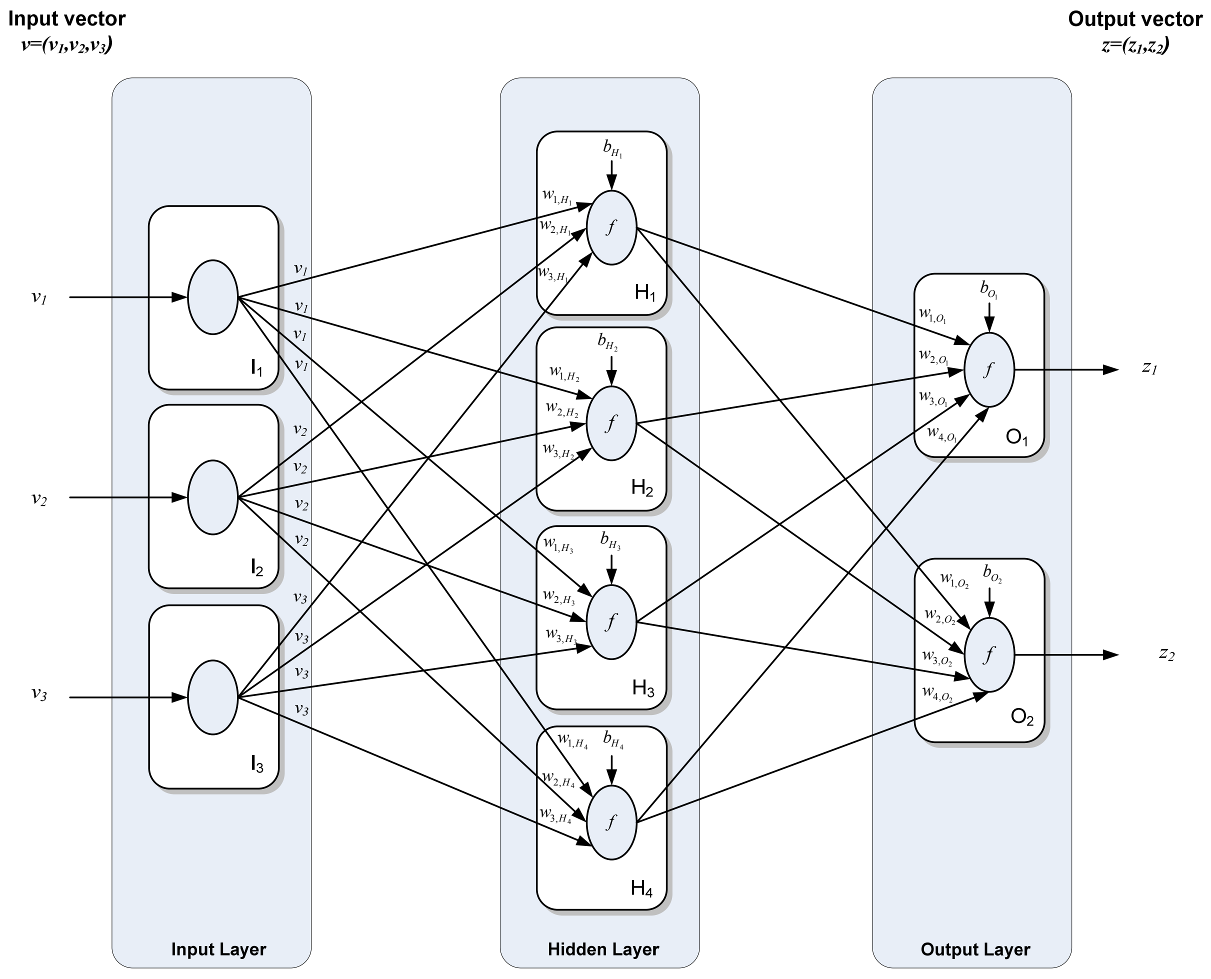


Figure 4.3 Example neural network, mapping 3 inputs to 2 outputs, featuring a single hidden layer consisting of 4 neurons

Before training begins all weights and biases of the network are initialised according to one of several initialisation methods – a simple one being random number generation within the range of -0.5 and 0.5 for each weight and bias. During training, the input part *v* of each sample in the training set is submitted as input to the network and the calculated output *z* is compared to the expected output *e*. The differences between the components of the latter two are used to update the configuration of the network accordingly. This update takes into account the difference between the expected and actual output and the first derivative of the transfer function in the form of the *delta* quantity. Some additional parameters also participate in the training process, namely, the *learning rate l,* that suggests how fast should the weights and biases change during training to compensate for the error, and the *momentum m,* that intuitively represents change amplification according to previous changes, independent of the current error. The adaptation of the weights and biases typically takes place in *reverse layer order,* starting from the neurons of the output level towards those of the input level. This due to the fact that the target output contained in each example practically refers to the output level of the network. Networks trained according to this policy are called *back-propagation* neural networks. Several functions exist for the combination of the aforementioned training parameters, an example given below:

- Delta $\delta_i = (e_i - z_i) \cdot f'(S_i)$ where the latter term signifies the first derivative of the transfer function for the current input
- New weight $w'_i = w_i + l \cdot \delta_i \cdot f(S_i) + m(w_i - `w_i)$ where the latter parenthesis signifies the difference between the current and the previous weight value.
- New bias $b'_i = b_i + l \cdot \delta_i + m(b_i - `b_i)$ where the latter parenthesis signifies the difference between the current and the previous bias value.

4.3.3 Genetic Algorithms

There is an extension to the *"neural networks are the second best way to do almost anything"* phrase opening the previous section that states *"...and genetic algorithms are the third"* [Russel02]. Regardless whether this is a generally accepted statement or not, it implies that, at least, genetic algorithms [Goldberg89], [Mitchell98], [Holland92] are valued as a powerful machine learning tool, comprising the most important member of the wider family of evolutionary search techniques. Hence, it comes as no surprise that the specific mechanisms, once again, have been based on an imitation of procedures originally found in nature. This time, it is the process of *natural selection* and *evolution* that is artificially reproduced, based on the *survival (and reproduction) of the fittest* among a *population* of representatives comprising a set of solutions to a specific problem or a set of examples of a specific concept.

Genetic problem solving typically has to deal with the evaluation and improvement of a population of candidate solutions or concept representatives. Intuitively, the algorithm's responsibility is to improve an initial solution population, by generating subsequent populations, characterised as *generations,* of gradually improved performance. Ideally, after a number of generations, the representatives contained therein will be of high, although often not optimal, performance. In algorithmic terms, genetic techniques are *hill-climbing* algorithms, aiming to approach any local or global extrema.

In order to achieve this, a number of tasks take place during the process of evolution, each serving a specific purpose in the overall mechanism. In particular,

Representation. Each solution or example is typically encoded as a binary string or a vector of integer or real values – the *chromosome*. The kind of representation chosen plays an important role in the subsequent stage where representatives are used to produce the population of the next generation. Depending on the nature of the problem and the specific instance of the mechanism, one representation may be more appropriate than the other.

Generation of the initial population. Although generation of a random set of individuals may seem adequate, usually it is required that the initial population contains diverse examples in order to ensure that a large part of the solution space will be covered by the subsequent search.

Fitness function. As soon as a population is available, its members have to be evaluated in order to distinguish the *fittest* representatives. The fitness function itself may range from a simple weighted sum of attribute values to evaluation mechanisms of increased complexity. As an example of the latter, it is often the case that a neural network is employed as a sophisticated fitness function, thus yielding hybrid methodologies combining both techniques.

Selection methods. Although the fitness function may concretely suggest the best performing representatives among the population, it is not necessarily implied that only these representatives should be selected in order to produce the subsequent generation. The reason is that doing so may restrict solution search around *local* extrema, thus preventing the algorithm from exploring alternative locations of the solution space which may lead to improved solution sets or even *global* extrema. Hence, several policies exist in order to efficiently select a subset of the current generation to be used for the task described next.

Crossover. The selected representatives are combined in pairs to produce the members of the next generation. The combination usually relies on mutual exchange of a certain parts of their representations. The rationale is that individuals of increased performance are more likely to be produced from the combination of high performing parents.

Mutation. As already mentioned, selecting the best performing individuals among a generation does not necessarily guarantee best overall performance since it may prevent the algorithm from efficiently exploring the solution space. This notion actually represents the *exploitation vs. exploration* trade-off: focusing on best performing members of the current population and rely on them for producing subsequent generations (exploitation) against putting less emphasis on the best individuals of the current population while trying to generate new ones, not necessarily related to them (exploration). Mutation is a technique that allows adjustment of the behaviour of the genetic mechanism between the two aforementioned extremes by randomly altering the representations of the best individuals combined to produce the new members. All alternative mutation methods share the common purpose of introducing new attribute values and, hence, new solution points that can not be found in the current population.

Elitism. Instead of selecting and mutating individuals in order to produce new members, the best performing ones are merely *copied* to the next generation. In a sense, this method corresponds to the exploitation extreme of the aforementioned trade-off and is usually combined with the other techniques for the production of the new population.

4.3.4 Support Vector Machines

Support Vector Machines [Boser92] form the implementation of a machine learning methodology based on Statistical Learning Theory [Vapnik68] and represent a highly active contemporary research field. Although the architecture of a support vector machine is very similar to that of a neural network the construction method differs significantly. The learning process of a neural network aims to reduce the *empirical risk* of the network while maintaining a fixed network architecture, which implies a fixed *confidence interval.* Support Vector Machines, on the other hand, aim to reduce the *confidence interval* while maintaining a fixed *empirical risk* [Vapnik98]. Support Vector Machines for classification are based on the idea of finding not *any* but *the optimal boundary hyperplane* for class separation. This optimality is achieved by selecting the hyperplane with *equal* and *maximum* distance from the closest representatives of each class. These representatives, used for the selection of the most appropriate separating hyperplane are the *support vectors*. In a sense, the problem of learning how to separate efficiently all available examples is reduced to that of learning how to separate efficiently the support vectors.

However, it is very often the case that the training examples are *not* linearly separable, at least when referring to a realistic problem, and, hence, the definition of a boundary hyperplane is not feasible in the space where the examples are defined. In order to overcome this, the examples are *mapped* to spaces of higher dimension in a non-linear way that ensures that classes will be linearly separable. The technique to achieve this is known as the *kernel trick* and the corresponding functions are the *kernel functions* [Burbidge01]. The latter take advantage of the observation that the data vectors appear only in inner products with weights during the calculations. Kernel functions offer the ability to calculate the inner products of the higher dimension representation of the data vectors. A typical example of a kernel function is the *polynomial*

$$K(a,w) = \langle a,w \rangle^{d} = \langle \phi(a), \phi(w) \rangle$$

i.e. the inner product of two vectors raised to an integer power is equal to the inner product of the higher dimension representation of these vectors.

To illustrate these concepts we provide a simplified example based on the XOR function simulation. Figure 4.4(a) shows a XOR function representation where the TRUE combinations are represented by white dots and the FALSE combinations are represented by black dots. No straight line, which is the special case of hyperplane in the 2-dimensional space, exists that can serve as a separation boundary between the TRUE and FALSE instances. We now apply the mapping to the 3-dimensional space through the aforementioned kernel function for *d=2*. We provide the analysis of the kernel function calculation, thus showing the mapping of the 2-dimensional vectors to the 3-dimensional space.

$$\begin{aligned} K(a,w) = \langle a,w \rangle^{2} = [(a_1,a_2)\cdot(w_1,w_2)]^2 = (a_1w_1 + a_2w_2)^2 = a_1^2w_1^2 + a_2^2w_2^2 + 2a_1w_1a_2w_2 = \\ = (a_1^2, a_2^2, a_1a_2\sqrt{2})\cdot(w_1^2, w_2^2, w_1w_2\sqrt{2}) = \phi(a)\cdot\phi(w) = \langle \phi(a), \phi(w) \rangle \end{aligned}$$

Hence, each 2-dimensional vector *a* is mapped to a 3-dimensional one according to *ϕ(a)*. By applying the specific mapping function to the four 2-dimensional points of Figure

4.4(a) we obtain the four 3-dimensional points of Figure 4.4(b). Notice that the 3-dimensional TRUE and FALSE combinations are, indeed, linearly separable.

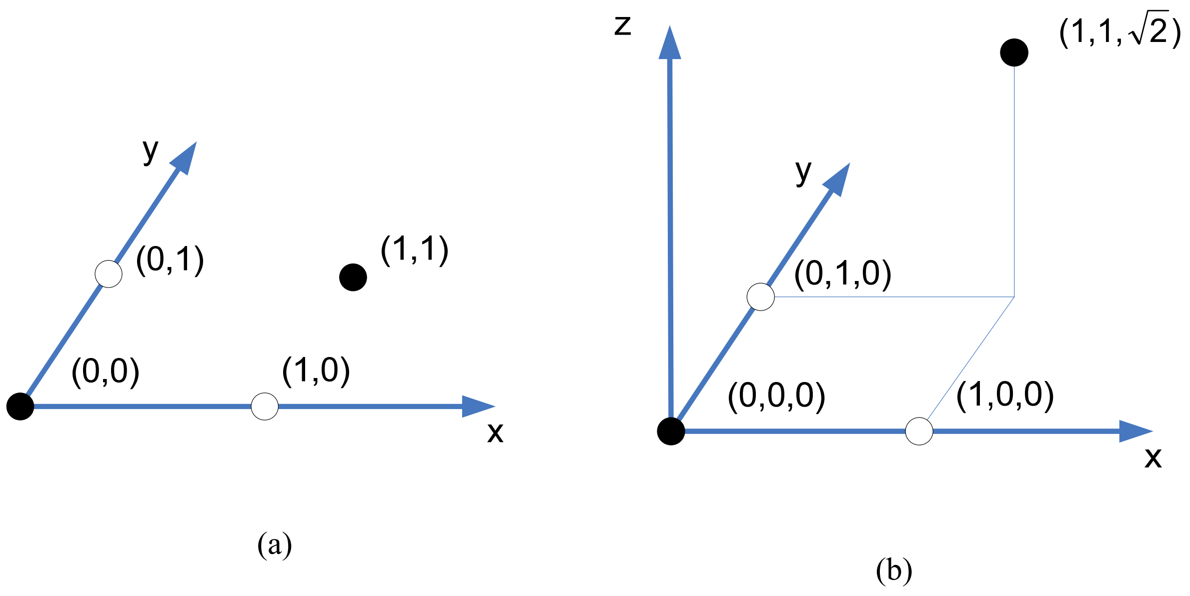


Figure 4.4 Four data vectors in the 2-dimensional space and their mappings in the 3-dimensional space according to mapping function ϕ

4.3.5 Latent Semantic Indexing and Multi-Dimensional Scaling

Latent Semantic Indexing (LSI) is a technique aiming to aid the semantic analysis of a large collection of documents and offer intelligent response to term-based queries upon these documents [Deerwester90],[Papadimitriou00]. It relies on the representation of each document as a multi-dimensional vector where each dimension element represents the presence, in the form of number of occurrences or absence, of one term. These terms exclude common words of reduced information value (*stop words)* and are usually represented by *stems* to avoid considering several forms of the same word as distinct terms. Due to the high dimension of this space, the technique employs Singular Value Decomposition (SVD) – a form of Principal Component Analysis – in order to map these vectors to a lower dimension space, while maintaining information regarding the relative *distance* between any two vectors. The idea is to capture and amplify the most important hidden relations between terms, thus improving the search outcome, while accelerating the search process and reducing the storage requirements by lowering the dimension of the search space. As a result, considering words appearing often in the same document, the outcome of a query containing only one of these words may include a document containing only the other. Nevertheless, the technique requires considerable pre-processing regarding the extraction of meaningful participating terms.

Multi-dimensional Scaling (MDS) [Takane77] or Perceptual Mapping is based on the same idea of reducing the space dimension. However, the aim in this case is to map multi-dimensional samples to 3-dimensional counterparts that can be visualised and, hence, intuitively inspected by humans.

4.3.6 Boosting

The fundamental idea of the Boosting technique is the employment of *weak learners,* i.e. machine learning mechanisms of low performance, in order to create a committee of much higher performance than any of its members [Schapire90],[Witten05]. [Schapire90] formally proved that any *weak learner*, i.e. an algorithm that may learn a class of concepts with accuracy slightly better than random guessing, can be recursively applied, making the error arbitrarily small, thus yielding a *strong learner*. The boosting technique practically applies the principles of incremental learning [Elman93] in the sense that the overall machine learning mechanism, i.e. the *committee,* is not of fixed size.

The technique takes advantage of the fact that a considerable amount of time during the training of a typical machine learning mechanism is spent for the fine tuning of its parameters in order to optimise its performance. As it is usually the case, this stage of adjustments contributes a relatively small error reduction, not proportional to the time required to achieve it. The Boosting approach, on the other hand, proposes the termination of the training of each one of its members as soon as its performance falls below a relatively high and quickly reached error margin. In a sense, each committee member is responsible of learning only a small *part* of the example set.

4.3.6.1 AdaBoost and Learn++

AdaBoost [Freund97] has set the foundation for a series of variations of boosting algorithms. In the following we briefly describe the standard approach and the modifications that have lead to the Learn++ variation [Polikar01],[Polikar02], which forms the basis of the algorithm we have proposed and implemented in the current context. The details of the algorithm applied in the current work are presented in the next chapter.

The input of the AdaBoost classifier consists of:

1. a set of N classified examples *{(x_1,c_1), ... , (x_N,c_N)},* where x_i represents an example and c_i its classification,
2. a distribution *D* over these examples,
3. a weak learning mechanism and
4. the number of iterations the weak learning mechanism will be invoked.

Each example is also characterised by a *weight,* which is updated at each iteration cycle and is intuitively analogous to the difficulty of the weak learning mechanism to classify the specific example correctly. This fact becomes apparent by the weight update rule presented below.

In particular, at the very beginning of the algorithm the weights are initialised explicitly based on *D.* For example, if *D* implies so, all examples start with equal weights. Next, during each iteration cycle, the steps appearing in Algorithm 4.1 take place:

```
Step 1.  Normalise the weights of the examples.
Step 2.  Train a new weak learner $h_t$ on the examples, according to the distribution
suggested by their weights.
Step 3.  Calculate $\varepsilon_t$ as the sum of the weights of the examples erroneously classified by
the current weak learner $h_t$.
Step 4.  Calculate $\beta_t=\varepsilon_t/(1-\varepsilon_t)$ and multiply weights of correctly classified examples by $\beta_t$.
```

Algorithm 4.1 The main body of the iteration of the AdaBoost algorithm

Notice that, by definition of the weak learning mechanism, $\varepsilon_t \in [0,½)$, i.e. it performs slightly better than random guessing. This implies that $\beta_t<1$ which, in turn, suggests that the weights of correctly classified examples are gradually reduced. Due to the training according to distribution, taking place in Step 2, the algorithm emphasises on hard-to-classify examples by constructing weak learners increasingly focused on them. After its construction, each weak learner *t* is characterised by a voting weight equal to $log(1/\beta_t)$ calculated at the end of its training. The output of the overall trained mechanism for any example is the result of *voting* of the weak learners. The class representing the weak learners' final voting decision for a specific example is the class for which the sum of voting weights is the highest.

The Learn++ algorithm applies the same principles in a more restricted context where not all examples are always available. Instead, examples are presented to the algorithm in groups, leading to the construction of sub-committees that have been trained based exclusively on a particular subset of the overall example set. In this context, every time a weak learner is added to the sub-committee, the entire committee is used to evaluate the current training subset to verify that the overall error does not exceed the threshold of ½. In a sense, Learn++ recursively applies the rationale of AdaBoost, both for weak learners with respect to the entire committee, as well as for the sub-committees with respect to the entire committee. The calculation of the overall error and the final voting decisions of the overall trained mechanism are exactly the same, thus maintaining the strong learning properties of the original approach.

4.4 Concluding Remarks

The context of the Open-MultiCAD design environment poses certain restrictions due to its functionality. These restrictions suggest corresponding requirements that had to be fulfilled by any mechanism employed to act as the Machine Learning Component of the proposed Intelligent User Profile Module. In particular, each user submits the declarative description of a scene and requests the generation of alternative geometric representations fulfilling this description. The user may subsequently inspect and evaluate these geometric representations at will, according to personal preferences, thus concluding a typical session of system use. Next time the user interacts with the system, he/she will submit another declarative description requesting the corresponding geometric representations. It becomes clear that, in the specific context, the feedback regarding the user's preferences, i.e. the training examples for any mechanism chosen to learn and subsequently apply these preferences, are produced in bursts, at different and distinct times. Moreover, the volume of the produced solutions may vary, sometimes being large thus prohibiting storage of all generated solutions for future use. Therefore, we can define a set of requirements for the machine learning mechanism to be employed in the current context. In particular:

1. It should be able to extend its knowledge regarding user preferences based on newly acquired examples.

2. It should not require previously seen examples in order to retain (at least up to an adequate degree) previously acquired knowledge.
3. It should be flexible, supporting a wide range of sizes for the example set.
4. It should not rely on similarity of examples for similarity in user preferences.

Most of the state-of-the-art mechanisms presented in the current chapter represent powerful methodologies, able to capture and simulate any pattern in a given set of examples. However, as is usually the case, there are certain trade-offs that prevent these mechanisms from fulfilling the aforementioned requirements. In particular, most of the methods presented rely on the entire example set in order to acquire an efficient model of the data. Newly acquired information has to be included in the overall example set and the mechanism has to be retrained with the entire population in order to achieve adequate performance. If only the new members are presented we encounter *catastrophic forgetting* where the mechanism learns the current examples but previous knowledge is lost. Hence, even if the first of the aforementioned requirements is fulfilled the second is not in the majority of cases. Moreover, mechanisms of fixed architecture and subsequent complexity may not respond equally well in variable example set sizes as demanded by the third requirement. Therefore, we may encounter cases where the mechanism's complexity is not analogous to the example set, given the fact that the generated solutions may range from a few hundreds to a few hundred thousands. Last but not least, since user preferences may be unpredictable, the employed mechanism should be flexible enough to accept very similar solutions to be classified differently. Hence, traditional statistical methods and unsupervised learning could not respond to the needs of the current context.

For the above reasons we have chosen to propose and implement a machine learning algorithm based on AdaBoost and its Learn++ variation using back-propagation feed-forward neural networks as weak learners. The details of the implementation are given in the next chapter. Here, we summarise the advantages of our approach in the following:

1. The proposed mechanism is able to incorporate new knowledge based on new examples while adequately retaining previously acquired knowledge.
2. Acquisition of new knowledge is achieved through the addition of new members to the Machine Learning Component committee only if necessary. In this way the complexity of

the mechanism is adapted to the complexity of the user preferences as suggested by the examples.

3. The proposed mechanism does not need to consult previously seen examples in order to retain previously acquired knowledge.
4. The proposed mechanism offers a degree of flexibility regarding the weak learning mechanism used.

In the following chapter we elaborate on the Machine Learning Component and the details of its implementation.

Part II

5 Intelligent User Profile Module

Based on the research and discussion presented in the previous chapters we describe in the current chapter the design and implementation of a Hybrid Intelligent User Profile Module for the Open-MultiCAD Design Environment. The proposed module consists of two separate, yet complementary components, each realising an alternative approach towards intelligent decision support with respect to user preferences in the context of object design.

The structure of the current chapter follows the architecture of the module. In particular, the chapter commences with a justification of the system architecture adopted by our module and continues with an overview of this architecture as proposed and implemented in the Intelligent User Profile Module. This overview also serves as an introduction to the nomenclature and the role of the prototype components of the implementation. Next, the set of fundamental assumptions that support the employed mechanisms and the integration of the latter to the already existing system are presented in detail. Each component is described thoroughly, regarding the underlying theoretical aspects and the respective algorithms. The integration of the two components to the Open-MultiCAD environment is subsequently presented. Finally, certain implementation issues are discussed, focusing on points that influence the system performance in terms of required processing time and storage space.

5.1 Rationale

Open-MultiCAD comprises a design framework that facilitates the transition from the abstract description of an object to its concrete visualised counterpart. The current stage of the Open-MultiCAD implementation is on the domain of Architecture, accepting descriptions of building *scenes*, and subsequently generating and visualising alternative geometric *solutions* corresponding to these scenes. As presented in the previous chapter, a typical Open-MultiCAD input of moderate complexity may generate thousands of solutions as output. These solutions are all formally valid, at least based on the declarative description submitted by the user as input. Nevertheless, it quickly becomes apparent that they are not all equally interesting for the user and there is no practical way for him/her to actually inspect the visualisations of all these solutions in order to select the best, i.e. those closest to his/her informal preferences, among them. One may argue that the user may check solutions only until he/she discovers an adequate number of satisfactory ones; hence, it is very unlikely that he/she will have to go through the complete set of results. However, even in this case, the problem of presenting the solutions to the user in a meaningful order – instead of randomly or sequentially – can be trivially reduced to the same problem we have dealt with in this work and that is described in the following.

A system destined to incorporate a computational model of user's preferences has to cover two distinct and equally important stages of this process. In particular,

User Preferences Acquisition. At this stage the system has to collect information based on user's feedback regarding his/her preferences. This information has to be formulated appropriately in order to be exploitable by the mechanisms of the next stage.

User Preferences Application. At this stage the system has to apply the information acquired during the previous stage for the user's benefit during regular system use.

Ideally, a system incorporating a computational model of user's preferences, regardless of the specifics of the mechanism, could benefit from this model during two *distinct* phases of the Declarative Modelling Design Process presented in the corresponding chapter. In particular,

Scene Generation. Applying user preferences at this stage would allow the system to generate only solutions complying with the input description as well as with the current user's modelled preferences.

Scene Understanding. Applying user preferences at this stage would not interfere with the solution generation. Thus, the system would generate all solutions complying with the input description. Subsequently, and due to the application of user preferences, the system would divide the solutions in classes of different *degrees of preference*, presenting them in descending preference order to the user.

We have chosen to concentrate our efforts on the latter phase; hence the main direction we have chosen to follow is that of intelligent solution evaluation as opposed to intelligent solution generation. There is more than one reason justifying this as the major step towards the capture and application of user preferences in the current context.

In particular, it quickly becomes apparent that the main user feedback revealing morphological user preferences originates from the properties of the *geometric representations of the solutions* that have been *approved by the user*. In other words, regardless whether intelligent solution generation or intelligent solution evaluation is the ultimate aim, one has to deal with the geometric properties of the produced solutions in order to extract the morphological user preferences. This, in turn, implies the processing of the geometric representation of solutions and the extraction of the values for one or more observed attributes as the first major step towards the capture of user preferences. Once this model is available it may be applied to the solution generation stage, as a set of additional constraints further restricting the declarative description [Bardis05], or to the solution evaluation stage, as a filter for the already generated solutions. Since the acquisition of user preferences had already set the focus on the geometric solution representation we have chosen to further elaborate on this representation by concentrating on the Scene Understanding stage of the Declarative Modelling Design Process cycle.

From the technical point of view, concentrating on the geometric solution representation allows for an open module and a user preference model that may be applied not only to solutions generated for other user's declarative descriptions within the same environment but also to geometric representations from other environments, given a certain degree of compatibility.

We may also mention as one of the important side-effects the fact that the chosen direction has lead to a less restrictive and more promising area of research. In particular, ideas from other areas could be tested and contribute to the current context and, alternatively, the insight gained from the present research might have interesting applications to other, not directly connected, research axes. Last but not least, as a technical advantage, we have managed to enrich the already existing, yet constantly changing, Open-MultiCAD environment with an *add-on* module that could provide user modelling with *minimal, if any at all,* intervention to the other modules of this environment. Dealing with the output of the system, i.e. the geometric representation of the solutions, has allowed us to work independently and experiment at will. As it will become clear throughout the rest of this chapter, the Intelligent User Profile Module we have proposed and implemented, requires no alterations of the existing Open-MultiCAD environment thus enhancing the overall system modularity and extensibility.

5.2 Module Overview

As already discussed in the previous section, the proposed Intelligent User Profile Module had to fit to the already existing MultiCAD system with minimal intervention to the latter. Moreover, it had to offer additional functionality, in the form of user preferences modelling, without considerable additional overhead for the user.

5.2.1 Requirements

Practically, we have set and subsequently fulfilled two fundamental requirements for our module:

Minimal modifications to the existing Open-MultiCAD environment. We work with the geometric representation of the solutions. The User Profile database forms a separate set of entities and relationships that requires no change to the existing Open-MultiCAD database. Both components – *Decision Support (DS)* and *Machine Learning (ML)* – are optional and easily activated from within the existing environment.

Minimal user overhead. We require the user to initialise his/her Decision Support profile only in order to provide instant automatic solution evaluation and only once. Subsequently, it is assumed that the user evaluates some, if not all, generated solutions for the scene(s) he/she submits. Therefore, we take advantage of this evaluation in order to transparently train the Machine Learning Component thus creating the corresponding ML profile. If a DS profile already exists, we compare the performance of the two profiles.

In particular, we have chosen to require only a *pre-processing* feedback, acquired only once by every user, yielding an initial *Decision Support (DS) User Profile* that serves as an evaluation mechanism for instant solution evaluation as well as reference for the *Machine Learning (ML) User Profile* which is gradually built during the normal use of the system. Both profiles are acquired by the corresponding DS and ML *components* that store this information in the *User Profile Database*. The latter forms an interconnected yet distinct part of the Open-MultiCAD database, already presented in the corresponding chapter. Every time the user goes through a session by submitting a scene description and evaluating the solutions produced, the two alternative profiles are also used to transparently evaluate the solutions. The performance of the two profiles is compared and as soon as the ML profile outperforms the DS profile the user is notified. Moreover, automatic solution evaluation is available from the very first session through the DS profile since the latter is based on pre-processing user feedback.

5.2.2 Module Operation

The data exchange among the entities of the integrated system appears in Figure 5.1. The only direct feedback the user has to provide to the Intelligent User Profile Module is the data for the initialisation of the DS profile. Subsequent feedback is retrieved through the regular solution evaluation that takes place during normal system use.

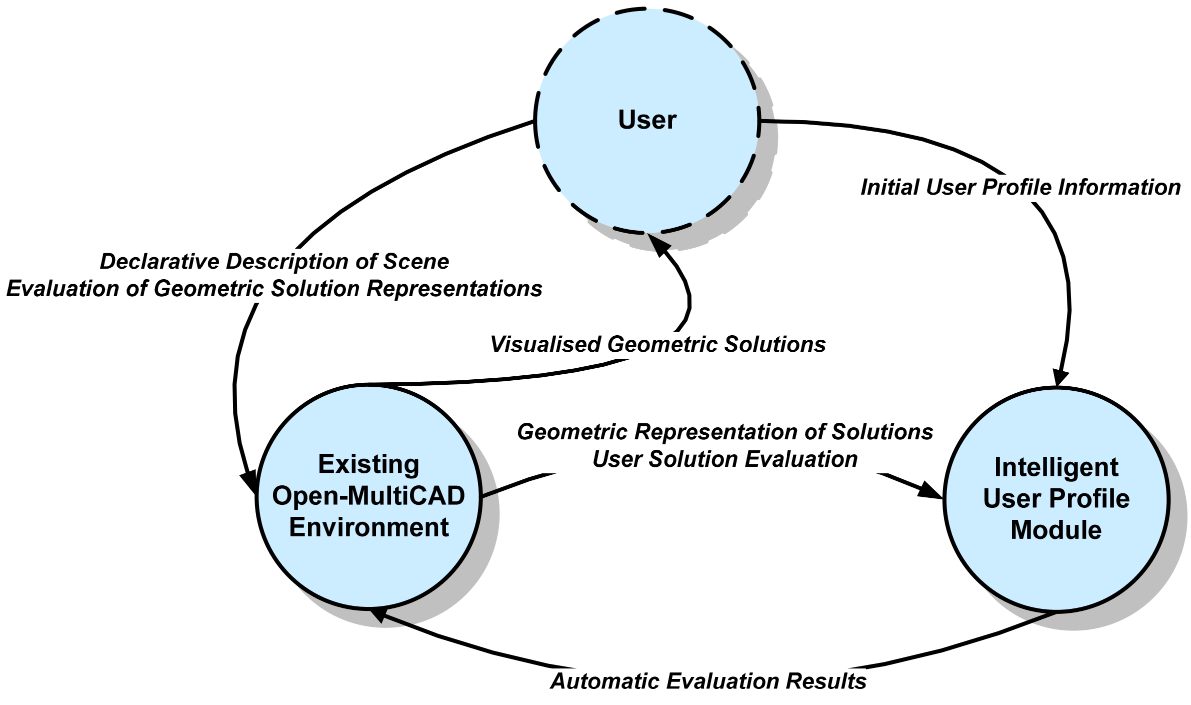

User
Initial User Profile Information
Declarative Description of Scene
Evaluation of Geometric Solution Representations
Visualised Geometric Solutions
Geometric Representation of Solutions
User Solution Evaluation
Existing Open-MultiCAD Environment
Intelligent User Profile Module
Automatic Evaluation Results

Figure 5.1 Data Exchange in User Profile MultiCAD

This is concretely depicted in Figure 5.2. The integration of the components comprising User Profile Module is shown therein as part of a typical scenario of system use.

The tasks carried out in the context of the already existing Open-MultiCAD system appear gray-shaded in this diagram. Notice how the typical *Open-MultiCAD cycle* remains in tact despite the additional module and the consequent data exchange. Contributing to this direction, Profile Initialisation is a task completely disconnected from the cycle, thus not directly affecting it in any way. Hence, after the Profile Initialisation has taken place, the specific user will have to provide no additional feedback during regular use of the system.

In order to clarify the module's operation we will assume in the following that the user has already completed a certain number of sessions after profile initialisation. This implies that there are a number of encoded solutions already available together with the corresponding user's evaluation for each one of them. This assumption will allow us to fully describe the operation and participation of the Machine Learning component to the automatic solution evaluation process. In particular, a typical session of system use starts with the submission of the declarative description of a scene that will be used as input to the Solution Generator. The user may request automatic solution evaluation for the results and may also be willing to manually evaluate the solutions. Automatic solution evaluation is available according to the Decision Support profile and the Machine Learning profile. The former has been defined during Profile Initialisation. The latter is gradually built and refined every time the user manually evaluates a solution population. Both profiles apply not to the geometric representation of each solution but to its encoded version as a vector of attribute values. These values represent each solution's performance with respect to certain observed morphological characteristics.

Hence, as soon as the solutions for the current scene are produced they are encoded and stored in the User Profile database, each coupled with a pointer to its geometric representation stored in the Open-MultiCAD database. Next, each solution is evaluated according to the Decision Support profile as well as the Machine Learning profile. Each profile yields an alternative grade for each solution that is also stored in the User Profile Database. In case the user has chosen not to proceed with the manual solution evaluation, the produced solutions are visualised and presented to him/her in descending order of automatically calculated preference. At the current stage of the implementation the overall calculated preference is assumed to be the average of the grades assigned by the alternative profiles.

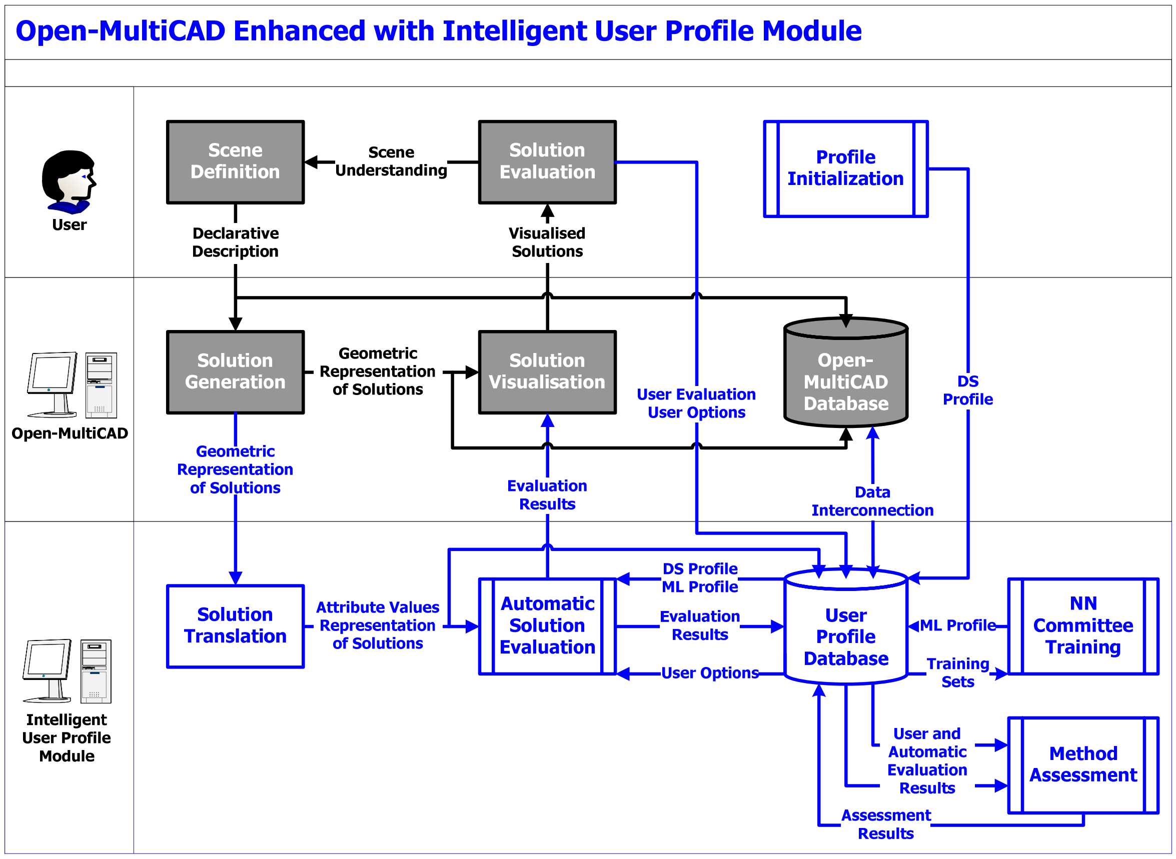


Figure 5.2 Typical Session of use of the Intelligent User Profile Module

In case the user decides to proceed with the manual solution evaluation the module takes advantage of the feedback in order to perform a number of tasks. In particular, the manually graded solution population is used in order to:

- Refine the Machine Learning Profile.
- Evaluate the consistency of the Decision Support Profile.
- Compare the performance of the alternative profiles.

Each encoded solution, coupled with its grade *provided by the user*, is used as part of a new *training set for the Machine Learning component.* Notice that the specific set of solutions has already been used to *evaluate* the Machine Learning profile up to the specific point. Hence, we are able to assess the improvement caused by the new training set to its performance. Furthermore, as it is discussed in detail in the next chapter, we are able to measure the effect of the newly acquired knowledge to the ability to classify *older* examples, already used for training.

Similarly, the user's grading is used to assess the credibility of the Decision Support profile. It may be the case that the user is not consistent, up to a certain degree, with the answers he/she has provided during the Decision Support profile initialisation. Last but not least, having the user's grading available, we are able to compare the performance of the alternative profiles with respect to the current scene as well as regarding the overall user's statistics.

No redundant information is stored in the User Profile database. Instead of that, the User Profile database is interconnected with the already existing Open-MultiCAD database through the required scenes and solutions ID's thus eliminating the need for data replication and the consequent need for consistency checking. Moreover, since certain tasks may require a large amount of time in order to be completed their results are immediately stored in the User Profile database thus offering the user the ability to postpone further processing. For example, solution translation may take place at one time and automatic solution evaluation at another time. This architecture allows the User Profile Module to handle scenes that lead to large numbers of generated solutions.

Automatic Solution Evaluation combines both components of the User Profile Module providing alternative evaluations for each solution as shown in Figure 5.3. Each solution is automatically evaluated according to the Decision Support Profile and the Machine Learning Profile given that the latter has already received training. In case the user is not willing to evaluate the specific solutions and has requested automatic solution evaluation, the output of the specific action is used during visualisation, in order to present solutions in descending preference order. The order is based on the automatically calculated degree of user preference with respect to the profiles the user has chosen to invoke for the current session.

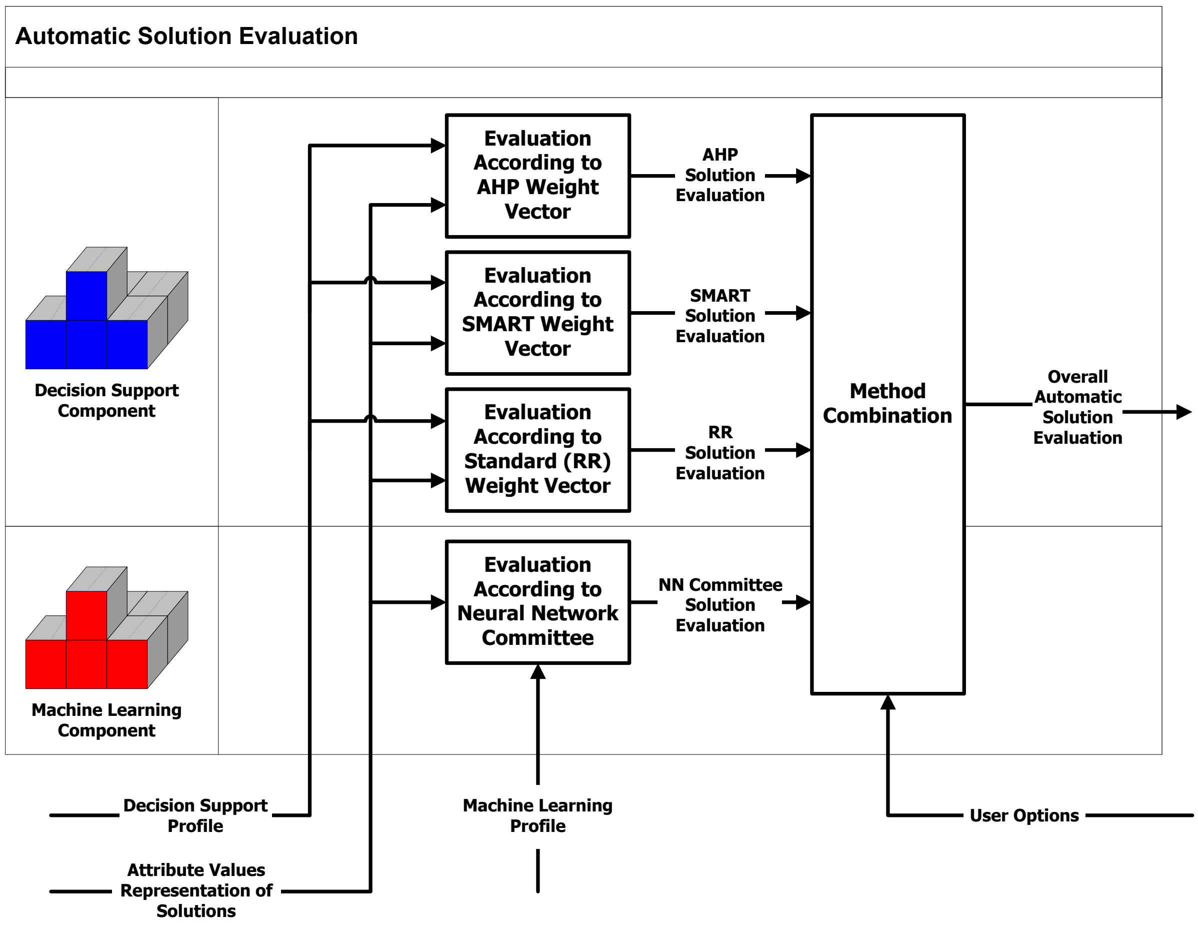

Automatic Solution Evaluation
Decision Support Component
Machine Learning Component
Evaluation According to AHP Weight Vector
Evaluation According to SMART Weight Vector
Evaluation According to Standard (RR) Weight Vector
Evaluation According to Neural Network Committee
AHP Solution Evaluation
SMART Solution Evaluation
RR Solution Evaluation
NN Committee Solution Evaluation
Method Combination
Overall Automatic Solution Evaluation
Decision Support Profile
Attribute Values Representation of Solutions
Machine Learning Profile
User Options

Figure 5.3 Automatic Solution Evaluation sub-tasks

5.3 Solution Encoding

Each solution produced by the Solution Generator module of MultiCAD consists of a set of bounding boxes signifying the corresponding spaces of the building. These spaces may be rooms, habitation zones or even sub-buildings depending on the declarative description submitted as input. The geometric model used for the bounding boxes is a rather simple one and has already been presented in chapter the Declarative Modelling chapter. Briefly presented, each area is represented by its origin, its dimensions, its colour/texture and two labels revealing the *geometric primitive* it is based upon (cylinder, box, etc.) and the *object type* (kitchen, bedroom, habitation, etc.) it represents. An extra label for the entire solution reveals its project type (habitation, hotel, office, etc.). This geometric model has been intentionally kept simple throughout the development of the Open-MultiCAD environment in an effort to facilitate future extensions and/or integration of alternative geometric representations to the existing system. We have conformed to this approach in its entirety and, therefore, we have based the solution encoding, described below, only on the aforementioned geometric representation, requiring no extra information for the spaces that would, in turn, demand knowledge of the corresponding declarative description. Our goal has been to create a module that would be extensible, in the sense that it would be feasible to apply, at the expense of some minor modifications, to geometric models not necessarily originating from a declarative description.

Even this representation, albeit simple, may lead to a few hundred values for the representation of a single solution. In order to reduce the complexity of each solution and, moreover, introduce the qualitative aspect required by the preference mechanisms employed, we have chosen to observe, for each solution, a specific set of morphological and geometric attributes. These *observed attributes*, as they will be mentioned for the remainder of this chapter, have been chosen mainly according to the corresponding building properties used in [Fri03], [Mak05] and [She01] and represent the lowest level of the *attribute tree* already discussed in the Preference Modelling and Decision Analysis chapter. We will assume that, for the current context, this tree, although by no means exhaustive with respect to building assemblies properties, fulfils completeness. In particular, we assume that no attributes concerning materials, cost, energy performance, etc. are of interest to the user, i.e. the DM, mainly due to the fact that relevant information is not adequately present in the geometric representation of the solutions produced by the Open-MultiCAD environment. Moreover, this is the minimum tree that includes all attributes of interest and the evaluation of each attribute is independent of the others therefore *decomposability* and *minimum size* are also fulfilled. Finally, the specific lower level attributes have been appropriately selected to offer *absence of redundancy* and *operationality*, i.e. their evaluations do not overlap and they may be extracted directly from the geometric solution representation, thus ensuring complete fulfilment of all requirements for a valid attribute tree.

The observed morphological attributes and their parameters vary for different kinds of buildings, i.e. different *project types*. However, they can always be extracted by the simple geometric model described before. Thus, each solution is *mapped* by the module to a *vector of values*, which represents the performance of the specific solution against the set of observed attributes. Both components of the User Profile module operate on this attribute encoded version of the solutions. For the rest of this chapter, we will refer to *encoded solutions* whenever it is necessary to distinguish from their geometric counterparts.

Depending on the project type, the observed attributes as well as their ranges vary. As an example, Table 5.1 summarises the observed attributes for the project type *residence.* The corresponding information for alternative project types can be trivially defined in the Open-MultiCAD database. Regarding the minima for the residence project type we have tried to capture extreme cases like *studios* where, for example, no explicit bedroom exists. The maxima, on the other hand, depend on the site dimensions and typically do not have to be fixed numbers. However, as it will be presented in detail in the next chapter, we have chosen to use similar site dimensions for each series of experiments in order to ensure comparability among different scenes submitted by the same user and their corresponding solutions.

Observed Attribute	***Symbol***	***Range***
Private zone area	PRA	0..maxprivatezone
Public zone area	PUA	0..maxpubliczone
Number of washrooms	NOW	1..maxwashrooms
Number of sleep rooms	NOS	0..maxsleeprooms
Non-Oblong rooms percentage	NOP	0%..100%
Private/Public zone separation	PPS	false,true
South-western bedroom	SWB	false,true

Table 5.1 Observed attributes for Residence project type

Each attribute is calculated from the elementary geometric properties of each space. Only spaces of the lowest level of the *part-of* hierarchical decomposition of the declarative description are considered at this stage. Practically, these are spaces that their object type falls within a pre-defined set that participates in the attribute calculation. Table 5.2 presents these object types all together, as in object type set *T*, as well as in groups, reflecting the needs of some of the corresponding observed attributes.

Object Type Set Description	***Object Type Set Symbol***	***Object Type Set Contents***
Terminal	T	{Bedroom, Bathroom, Corridor, Dining Room, Garage, Guest, Hall Room, Kitchen, Living Room, Office, Play Room, Storage, WC}
Public Zone	PU	{Corridor, Dining Room, Hall, Guest Room, Living Room, Play Room, WC}
Private Zone	PR	{Bedroom, Bathroom, Kitchen, Office, Storage}
Sleep Room	SL	{Bedroom, Guest Room}
Washroom	WA	{Bathroom, WC}
Potential non-oblong	PNOB	{Bedroom, Bathroom, Dining Room, Guest Room, Kitchen, Living Room, Play Room, Office}

Table 5.2 Object types and grouping according to attributes

It holds that

$$PU \subset T,\ PR \subset T,\ SR \subset T,\ WA \subset T,\ PNOB \subset T$$

Notice that it may be the case that certain object types belong to more than one category. For example, bedrooms participate in the calculation of the overall private zone area and are also considered when checking the building for the percentage of non-oblong rooms.

The following sections outline the specifics of the attribute values calculation for each solution. Some require a simple arithmetic operation whereas others an extensive search among all participating object types. In order to provide a concrete definition for each attribute we will consider each solution *S* as a set of objects. Therefore, for any object *o* mentioned in the following it will be implied that $o \in S$. Moreover, according to the aforementioned sets of objects types, for any object $o \in S$ we will use the notation *ot(o)* to signify its object type.

5.3.1 Public Zone Area, Private Zone Area

Intuitively, the private zone area of a solution is the sum of the areas of all spaces of bedroom, bathroom, kitchen, office and storage object type. The corresponding sum is calculated for the public zone object types. Formally for any solution *s*,

$$PublicZA_s = \sum_{ot(o)\in PU} area_o \qquad PrivateZA_s = \sum_{ot(o)\in PR} area_o$$

where *area* represents the area of an object as calculated by its dimensions already contained in the geometric representation of the solution. For example, for an object *o* that is an instance of the geometric primitive *box* we have

$$area_o = l_o \cdot w_o$$

l representing the length and *w* the width of the box.

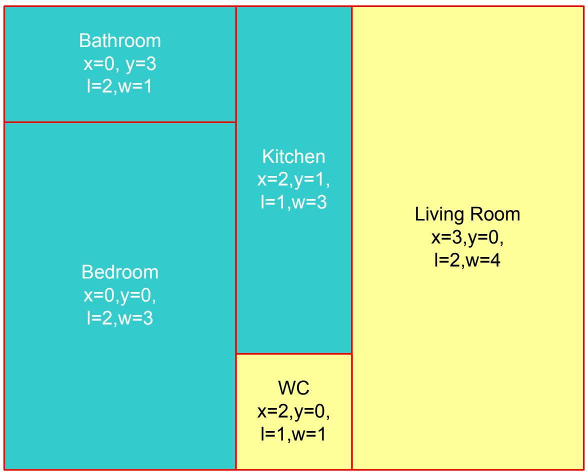
5.3.2 Number of Sleeping Rooms, Number of Washrooms

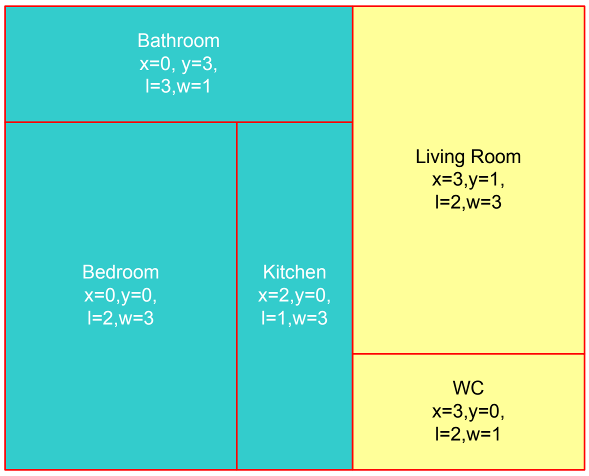
The number of objects of bedroom or guest room object type appearing in the solution contributes to the total number of bedrooms. Similarly, the number of objects of bathroom or WC object type appearing in the solution contributes to the total number of bathrooms. Formally,

$$NofSleepRooms_s = |o:ot(o) \in SR|$$

$$NofWashRooms_s = |o:ot(o) \in WA|$$

Although this information can be easily extracted by the declarative description of the *scene* that has been used as input for the generation of the solutions examined, we have chosen to include it in the observed attributes and extract it from the geometrical representation for a couple of reasons already outlined in the Rationale section above. In particular, we wanted our module to be able to directly compare solutions originating from alternative declarative descriptions without requiring any information from the declarative description itself, thus keeping the architecture open to geometric representations originating from alternative sources.





Solution A.
Public/Private Zone Separation is false because
$Max(x_{opr}+l_{opr})=3 \leq Min(x_{opu})=2$ *is false*
$Max(y_{opr}+w_{opr})=4 \leq Min(y_{opu})=0$ *is false*
$Max(x_{opu}+l_{opu})=4 \leq Min(x_{opr})=0$ *is false*
$Max(y_{opu}+w_{opu})=4 \leq Min(y_{opr})=0$ *is false*

Solution B
Public/Private Zone Separation is true because
$Max(x_{opr}+l_{opr})=3 \leq Min(x_{opu})=3$ *is true*
$Max(y_{opr}+w_{opr})=4 \leq Min(y_{opu})=0$ *is false*
$Max(x_{opu}+l_{opu})=4 \leq Min(x_{opr})=0$ *is false*
$Max(y_{opu}+w_{opu})=4 \leq Min(y_{opr})=0$ *is false*

Figure 5.4 Two solutions leading to different values of the public/private zone separation attribute

5.3.3 Public/Private Zone Separation

For an intuitive description of this attribute we may state that it is true when it is possible to draw a horizontal or vertical line across the top view of the building that would separate entirely the private zone objects from the public zone objects. The mathematical interpretation of this intuitive description with respect to the origin and the dimensions of each room is that the following logical statement is true:

$$max(x_{opr}+l_{opr}) \leq min(x_{opu}) \vee max(y_{opr}+w_{opr}) \leq min(y_{opu}) \vee$$

$$max(x_{opu}+l_{opu}) \leq min(x_{opr}) \vee max(y_{opu}+w_{opu}) \leq min(y_{opr})$$

where $ot(opr) \in PR$ and $ot(opu) \in PU$ and $opr,opu \in S$.

In the specific expression we have assumed that all objects are *boxes*. Practically, the expressions in the parentheses of the first row represent the east, west, north and south extremes of an object respectively. The latter is clearly depicted in Figure 5.4 where the dark objects represent the private zone and the light objects the public zone. The same solutions, as produced by the prototype appear in Figure 5.5.

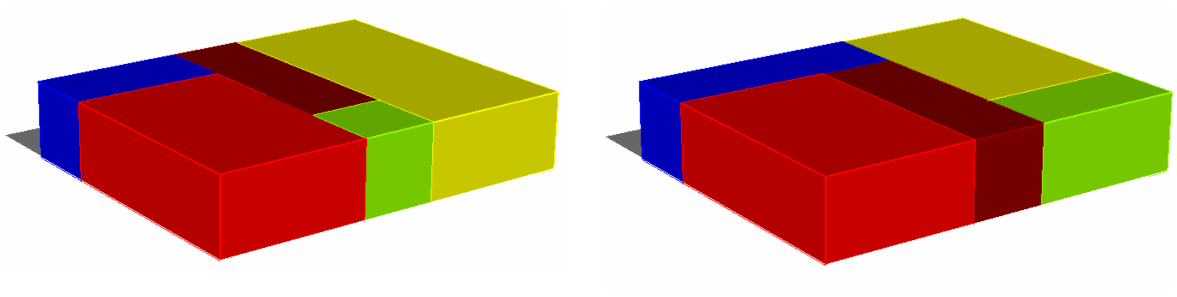


Figure 5.5 The two solutions of Figure 5.4 as produced by the system

5.3.4 Non-oblong Room Percentage

For the calculation of this attribute we consider the *PNOB* subset of the object types available for the definition of a declarative description and not the whole set of terminal object types *T* appearing in Table 5.2. The reason is that some object types are *inherently* oblong as, for example, in the case of the corridor. In other words, since we consider oblong rooms a disadvantage in the context of the specific attribute, it would be unfair for a solution to be downgraded because of the presence of an oblong corridor. Therefore, the object types we have chosen to consider in the calculation of this attribute are the ones appearing in the corresponding row of Table 5.2. Again concentrating on *box* instances, an object *o* is considered non-oblong when

$$o \ is \ non\text{-}oblong \Leftrightarrow \frac{l_o}{w_o} \geq 0.5 \ \ \wedge \ \ \frac{w_o}{l_o} \geq 0.5$$

where $ot(o) \in PNOB$. The attribute value for a specific solution is the percentage of objects for which the above condition is true, among objects of the aforementioned object types. Formally,

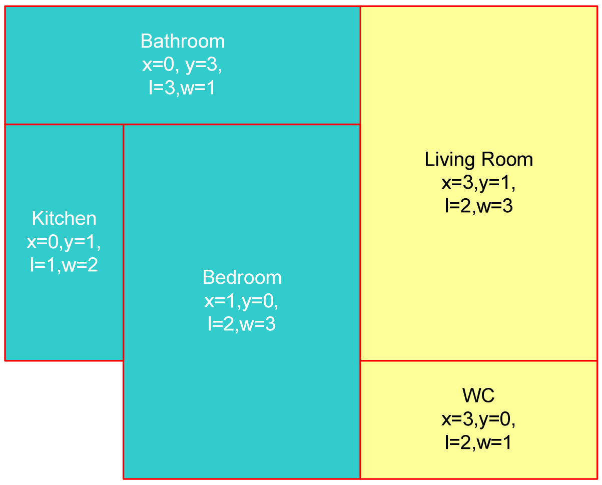
$$NonObPercentage = \frac{|nob : nob \ is \ non\text{-}oblong \wedge ot(nob) \in PNOB|}{|\, o : ot(o) \in PNOB \,|}$$

where *o,nob*∈*S*. Considering the solutions A and B appearing in Figure 5.4, we can easily calculate *NonObPercentageA*=80% and *NonObPercentageB*=60% since both solutions contain an oblong kitchen but Solution B also contains an oblong bathroom in a total of 5 objects of terminal object type.

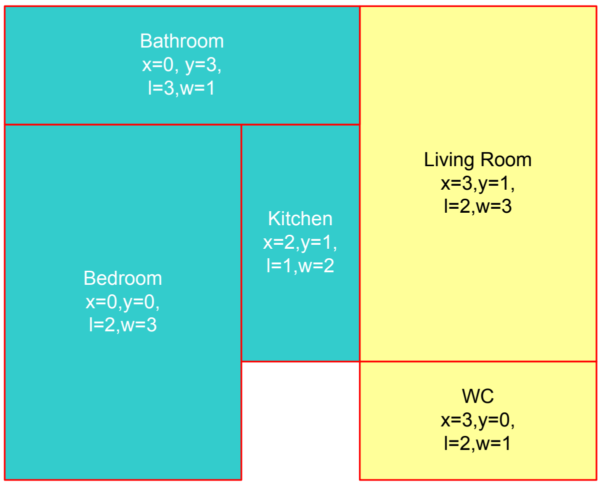
5.3.5 Southwestern Bedroom

This is also a binary attribute that may be true or false depending on the existence of a southwestern bedroom. Intuitively, we consider a solution to bear this feature when there is a bedroom having one of its walls completely exposed to the south and one wall completely exposed to the west. Formally, the observed attribute *Southwestern Bedroom* is true for a solution when

$$\exists b \in S : ot(b) = Bedroom \text{ such that}$$
$$\forall o \in S : ot(o) \in T,$$
$$(x_o + l_o \leq x_b \vee x_o \geq x_b + l_b \vee y_o \geq y_b + w_b) \wedge (y_o + w_o \leq y_b \vee y_o \geq y_b + w_b \vee x_o \geq x_b + l_b)$$



Solution A. Southwestern Bedroom is false because for the kitchen k
$x_k+l_k=1 \leq x_{be}=1$ is true
and
$y_k+w_k=3 \leq y_{be}=0$ is false and
$y_k=1 \geq y_{be}+w_{be}=3$ is false and
$x_k=0 \geq x_{be}+l_{be}=3$ is false



Solution B. Southwestern Bedroom is true For example, considering the kitchen k
$x_k+l_k=3 \leq x_{be}=1$ is false but
$x_k=2 \geq x_{be}+l_{be}=2$ is true
and
$y_k+w_k=3 \leq y_{be}=0$ is false and
$y_k=1 \geq y_{be}+w_{be}=3$ is false but
$x_k=2 \geq x_{be}+l_{be}=2$ is true

Figure 5.6 Two solutions that lead to different values of the south-western bedroom attribute

Two inequalities are required in order to discover an overlap parallel to an axis and a third inequality is used to eliminate the case of a “harmless” (north or east) overlap, hence the six inequalities in the logical condition. The operation of this logical condition is depicted in Figure 5.6. In particular, in Solution A, the first clause of the first parenthesis of the condition is *true,* thus eliminating the case of an overlap along the south (or north) wall, i.e. parallel to the *xx’* axis. Regarding the *yy’* axis the two *false* clauses in the second parenthesis reveal an overlap. The last clause of the second parenthesis makes the actual difference between the two solutions. The overlap, as far as the *yy’* axis is concerned is practically the same; nevertheless, solution B bears a “harmless” overlap along the east wall of the bedroom whereas in Solution A the overlap is on the west side of the bedroom.

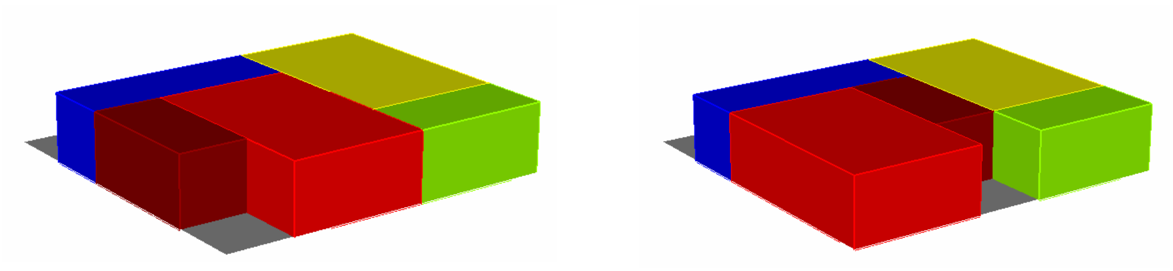


Figure 5.7 The solutions of Figure 5.6 as produced by the system

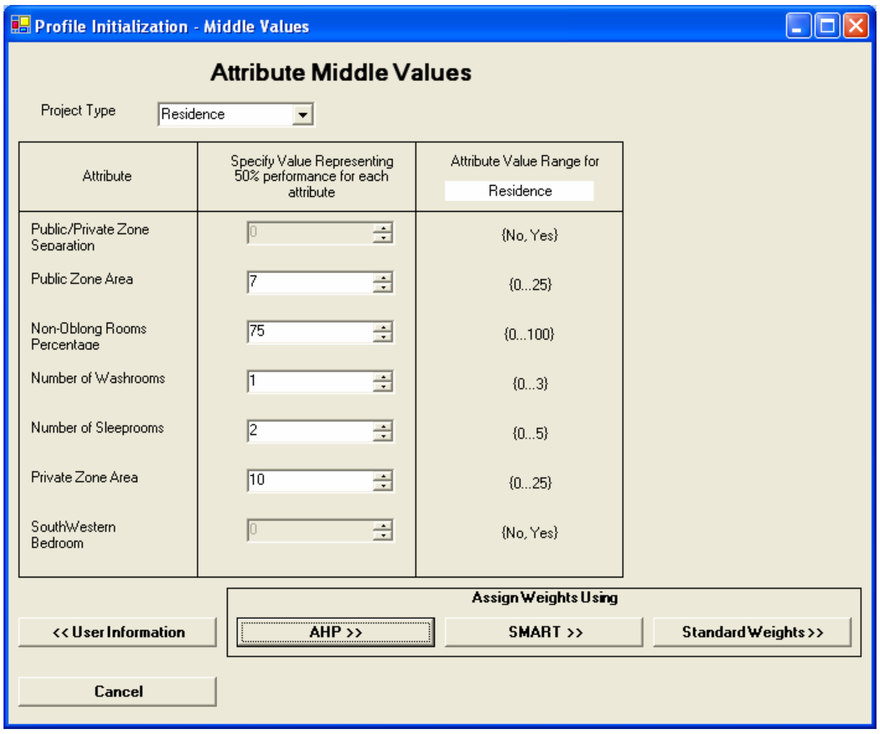
5.4 Middle Values

The range of values for each observed attribute varies depending on the project type. Each attribute value extracted by the geometric representation of a solution is subsequently normalised to a value between 0 and 1 representing the minimum and the maximum respectively. In order to increase accuracy during the creation of the user’s profile, we have incorporated in the attribute normalisation the notion of *bisection.*

In particular, it may be the case that the *arithmetic mean* of the range of an attribute does not represent a normalised performance of 0.5 according to a specific user. Therefore, the user is allowed to define a custom *middle value* for each attribute representing this 50% performance. For example, although the private zone area ranges from 0 to 25 in the example of Figure 5.8, the specific user has decided that value 7 represents the middle value for a solution with respect to the specific attribute. Hence, normalisation actually takes place in two distinct sub-ranges of the overall attribute value range. This implies that the *grading* of each solution *per attribute* does not depend only on its observed attribute values but also on the middle values chosen by the user. Formally, the *attribute grade* g_i the solution s receives for the observed attribute i is given by the formula:

$$g_i = \begin{cases} \dfrac{s_i - mid_i}{2(max_i - mid_i)} + 0.5, & max_i \geq s_i \geq mid_i \\ \dfrac{s_i - min_i}{2(mid_i - min_i)}, & min_i \leq s_i < mid_i \end{cases}$$

where s_i is the non-normalised value of attribute i for the specific solution, and min_i, mid_i, max_i, the minimum, middle and maximum values for the specific attribute respectively. We have chosen to assign the case of $s_i{=}mid_i$ to the upper branch to avoid problems of division with zero when the minimum and middle values are both zero, never allowing the maximum and middle values to be the same.


Profile Initialization - Middle Values

Attribute Middle Values

Project Type: Residence

Attribute	Specify Value Representing 50% performance for each attribute	Attribute Value Range for Residence
Public/Private Zone Separation	0	{No, Yes}
Public Zone Area	7	{0...25}
Non-Oblong Rooms Percentage	75	{0...100}
Number of Washrooms	1	{0...3}
Number of Sleeprooms	2	{0...5}
Private Zone Area	10	{0...25}
SouthWestern Bedroom	0	{No, Yes}

Figure 5.8 Middle Values User Interface

Notice that the minimum and maximum values depend only on the project type, whereas the middle value is custom, defined by the user. For example, according to the values appearing in Figure 5.8, a solution bearing total Public Zone Area equal to 11 units would receive a grade of

$$g_{PZA} = \frac{11-7}{25-7} + 0.5 \approx 0.72$$

according to the specific user's input, although 11 represents less than the middle of the acceptable range of values for the specific attribute. The reason is that the user has suggested that 7 units represents the middle value thus, intuitively, 11 units represent a more than adequate performance. This is exactly the kind of intuition we have tried to capture with the use of the bisection principle in the range of values for the observed attributes. In a sense, we can state that the notion of middle values offers a *vertical* evaluation, among values of the same attribute, whereas the notion of attribute weights offer a *horizontal* evaluation, among values of different attributes.

5.5 Decision Support Component

This is the first of the two components comprising the Intelligent User Profile Module, the other being the Machine Learning Component presented later in the current chapter. For the Decision Support component we have applied the principles for user preferences modelling that have been presented in chapter covering Preference Modelling and Decision Analysis. In particular, each user's preferences are modelled as a vector of weights representing the importance of each one of the observed attributes. Using the symbols of Table 5.1 for the observed attributes and w_i for the weight of the attribute i for a specific user u we have:

$$P_u=(w_{PRA}, w_{PUA}, w_{NOW}, w_{NOS}, w_{NOP}, w_{PPS}, w_{SWB})$$

In the following, and because of the construction method followed for the weight vector, it will always hold that, for a specific user,

$$\sum_{i \in A} w_i = 1$$

where *A={PRA,PUA,NOW,NOS,NOP,PPS,SWB}*, i.e. the weights revealing a user's preferences will always be normalised.

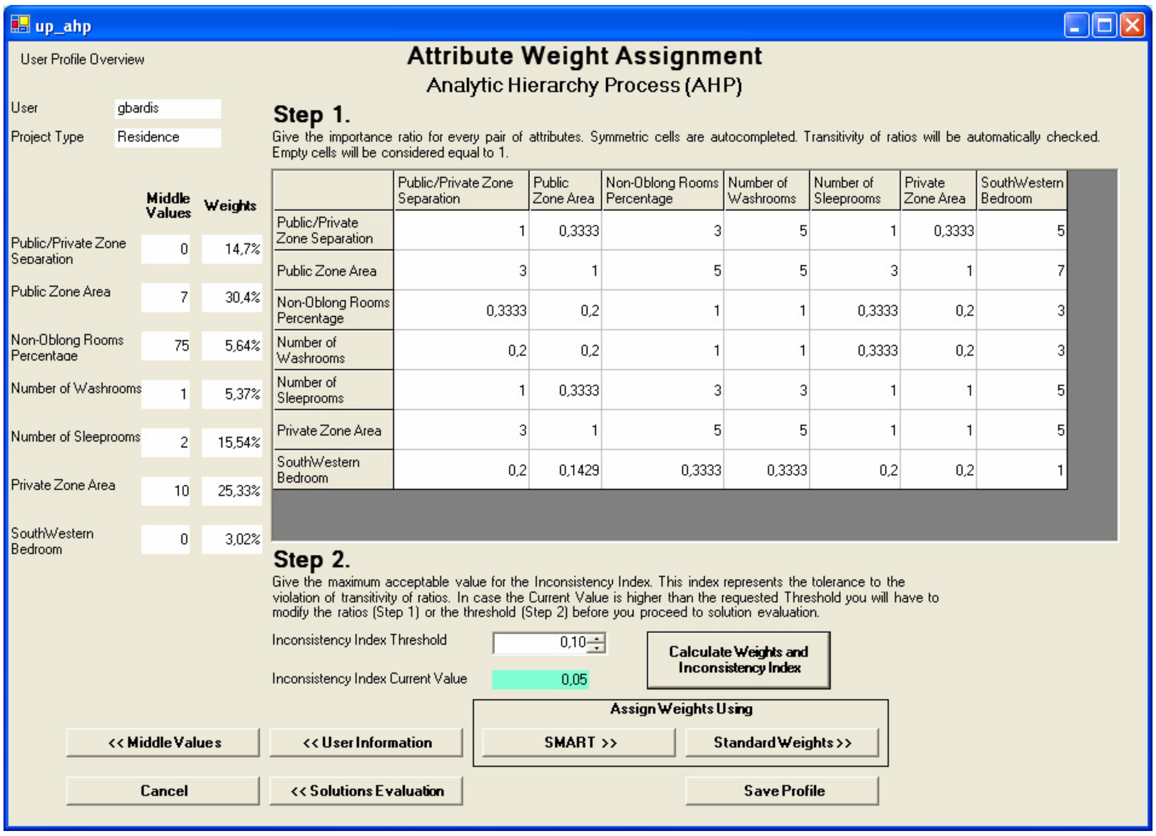
5.5.1 Decision Support Profile Initialisation

In order to enhance our module's ability to capture user preferences we have chosen to employ not a single but three alternative methods for this purpose. These methods originate from the decision support mechanisms already presented in the Preference Modelling and Decision Analysis chapter. Applying three alternative mechanisms offers flexibility and reliability to our module. In particular, flexibility stems from the fact that the user is able to select the method(s) that suits him/her best in expressing his/her preferences and still be able to take advantage of the module for intelligent decision support. The method selected can make a difference for the user since the three methods vary significantly regarding the input they require both in terms of quality and quantity. On the other hand, using all three methods – which is by no means obligatory – improves the reliability of the module as it will become apparent later when the Machine Learning Component is presented. We briefly describe below the three methods used for attribute weight assignment since they have already been presented in detail in the Preference Modelling and Decision Support chapter.

5.5.1.1 AHP Weight Assignment

This method requires the most detailed input on behalf of the user among the three weight assignment methods we have used. We have chosen to simplify the procedure, while maintaining the underlying principle of the method, by assuming that there exists only one general attribute that contains all aforementioned observed attributes. Moreover, because of the large number of alternative options represented by the generated solutions, it is practically impossible for the user to provide pair-wise comparison feedback for every pair of solutions. Therefore, the user has to provide feedback only in one pair-wise comparison table with respect to the observed attributes.

According to the method already described in detail in the Preference Modelling and Decision Analysis chapter, the user has to fill a table containing one row and one column for each observed attribute. Every cell of the table accepts the user's input regarding the *ratio of importance* between the two attributes of the respective row and column that intersect at the specific cell. The diagonal of this table contains 1's since it represents the ratio of importance for the same attribute. Moreover, the user is required to fill only half of the table – below or above the diagonal – since the product of symmetric cells is always 1. Remember that the user *does not have to* maintain transitivity when completing the table. The input may be contradicting in the sense that, the ratio for any two attributes is not necessarily the ratio implied by the product of other related ratios. For example, according to user's input in Figure 5.9, Public Zone Area is 3 times as important as Public/Private Zone Separation (second row, first cell). Public/Private Zone separation is, in turn, 3 times as important as Non-Oblong Rooms Percentage (first row, third cell). This could imply that Public Zone Area should be 9 times as important as Non-Oblong Rooms Percentage. However, the user was free to input that the latter ratio is only 5 (second row, third cell). This kind of inconsistency is acceptable by the method and it is measured by the Inconsistency Index. If the overall Inconsistency Index is acceptable, i.e. below a specific threshold, then the user's input is accepted as valid and the corresponding – normalised – weights are extracted from the table.


up_ahp

User Profile Overview

Attribute Weight Assignment

Analytic Hierarchy Process (AHP)

User	gbardis
Project Type	Residence

	Middle Values	Weights
Public/Private Zone Separation	0	14,7%
Public Zone Area	7	30,4%
Non-Oblong Rooms Percentage	75	5,64%
Number of Washrooms	1	5,37%
Number of Sleeprooms	2	15,54%
Private Zone Area	10	25,33%
SouthWestern Bedroom	0	3,02%

Step 1.

Give the importance ratio for every pair of attributes. Symmetric cells are autocompleted. Transitivity of ratios will be automatically checked. Empty cells will be considered equal to 1.

	Public/Private Zone Separation	Public Zone Area	Non-Oblong Rooms Percentage	Number of Washrooms	Number of Sleeprooms	Private Zone Area	SouthWestern Bedroom
Public/Private Zone Separation	1	0,3333	3	5	1	0,3333	5
Public Zone Area	3	1	5	5	3	1	7
Non-Oblong Rooms Percentage	0,3333	0,2	1	1	0,3333	0,2	3
Number of Washrooms	0,2	0,2	1	1	0,3333	0,2	3
Number of Sleeprooms	1	0,3333	3	3	1	1	5
Private Zone Area	3	1	5	5	1	1	5
SouthWestern Bedroom	0,2	0,1429	0,3333	0,3333	0,2	0,2	1

Step 2.

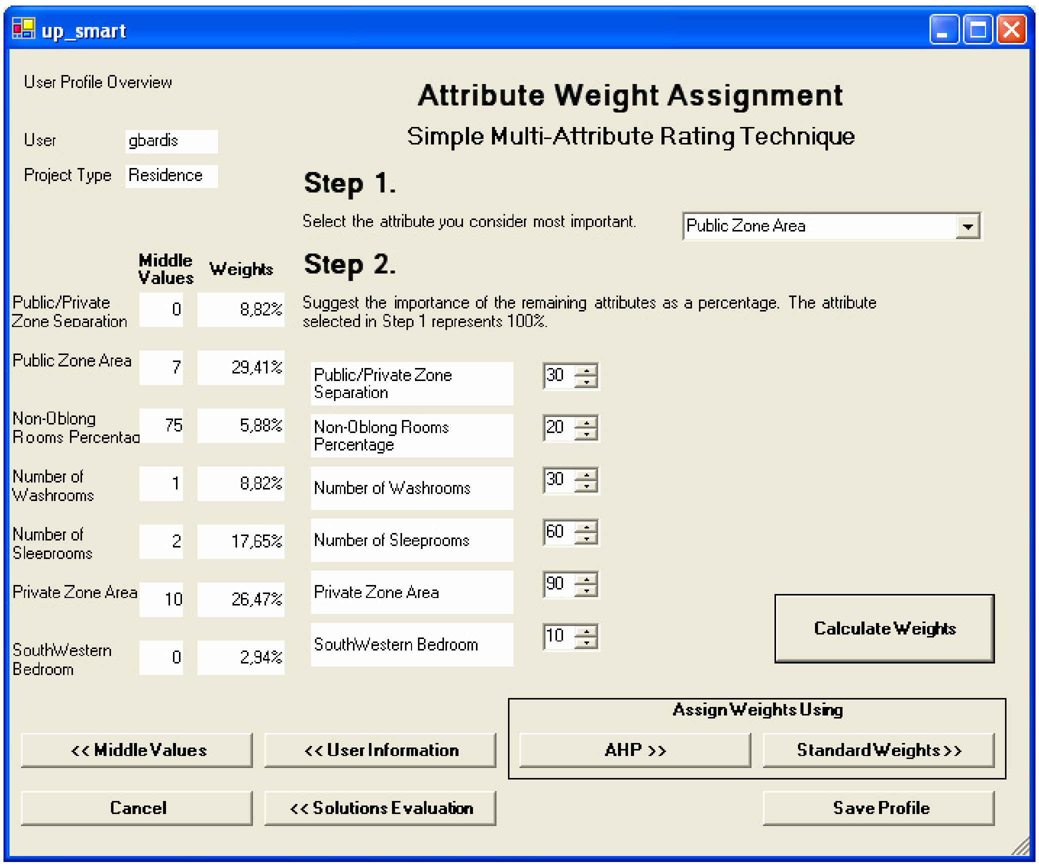
Give the maximum acceptable value for the Inconsistency Index. This index represents the tolerance to the violation of transitivity of ratios. In case the Current Value is higher than the requested Threshold you will have to modify the ratios (Step 1) or the threshold (Step 2) before you proceed to solution evaluation.

Inconsistency Index Threshold	0,10
Inconsistency Index Current Value	0,05

Figure 5.9 Attribute Weight Assignment according to Analytic Hierarchy Process (AHP)

5.5.1.2 *SMART Weight Assignment*

This method stands in the middle of the three methods employed for attribute weight assignment regarding the requirements for user input. In particular, the user here has to suggest the most important attribute, representing the 100% of importance. Next, he/she has to assign the importance percentage to each one of the other attributes when compared with the most important one. Typically these percentages will be less than 100%. The percentages assigned to each attribute by the user are then normalised to yield the corresponding weights. An example case of user input appears in Figure 5.10.



User Profile Overview

User	gbardis
Project Type	Residence

	Middle Values	Weights
Public/Private Zone Separation	0	8,82%
Public Zone Area	7	29,41%
Non-Oblong Rooms Percentag	75	5,88%
Number of Washrooms	1	8,82%
Number of Sleeprooms	2	17,65%
Private Zone Area	10	26,47%
SouthWestern Bedroom	0	2,94%

Step 1.

Select the attribute you consider most important. Public Zone Area

Step 2.

Suggest the importance of the remaining attributes as a percentage. The attribute selected in Step 1 represents 100%.

Public/Private Zone Separation	30
Non-Oblong Rooms Percentage	20
Number of Washrooms	30
Number of Sleeprooms	60
Private Zone Area	90
SouthWestern Bedroom	10

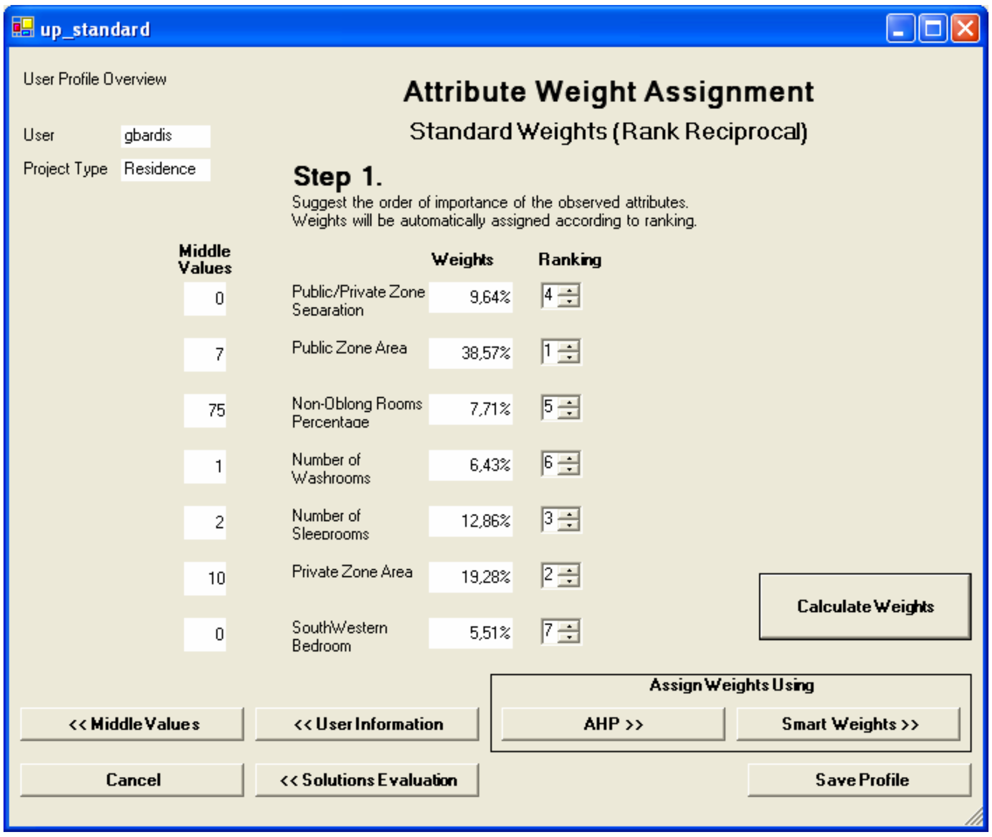
Figure 5.10 Attribute Weight Assignment according to Simple Multi-Attribute Rating Technique (SMART)

5.5.1.3 *Standard Weight Assignment*

Finally, this method requires the minimum input on behalf of the user among the three alternatives since the only information he/she has to contribute is the *order* of importance of the observed attributes and no weights whatsoever. The descending order submitted by the user suggests the *rank* of importance of each attribute and automatically implies its weight. The weights are predefined numbers based exclusively on the ranks and the total number of attributes. In particular, if attribute *i* has been selected as the most important one then its rank is 1, denoted as R_i=1. The formula for the weight of any attribute *i* is

$$w_i = \frac{1}{R_i \cdot \sum_{k=1}^{|A|} \frac{1}{k}}$$

where A is the set of observed attributes. This is the Rank Reciprocal method [Roberts02] presented in the Preference Modelling and Decision Support chapter. It is trivial to show that the weights produced by this method are already normalised. The user interface for this method appears in Figure 5.11.



User	gbardis
Project Type	Residence

Step 1.
Suggest the order of importance of the observed attributes.
Weights will be automatically assigned according to ranking.

Middle Values		Weights	Ranking
0	Public/Private Zone Separation	9,64%	4
7	Public Zone Area	38,57%	1
75	Non-Oblong Rooms Percentage	7,71%	5
1	Number of Washrooms	6,43%	6
2	Number of Sleeprooms	12,86%	3
10	Private Zone Area	19,28%	2
0	SouthWestern Bedroom	5,51%	7

Figure 5.11 Attribute Weight Assignment according to Rank Reciprocal method

5.5.2 Automatic Solution Evaluation

As it has become apparent, each one of the three alternative weight assignment methods leads to an alternative overall grade for each solution. The overall solution grade *b* is calculated as the inner product of its attribute grade vector with the appropriate weight vector. Formally,

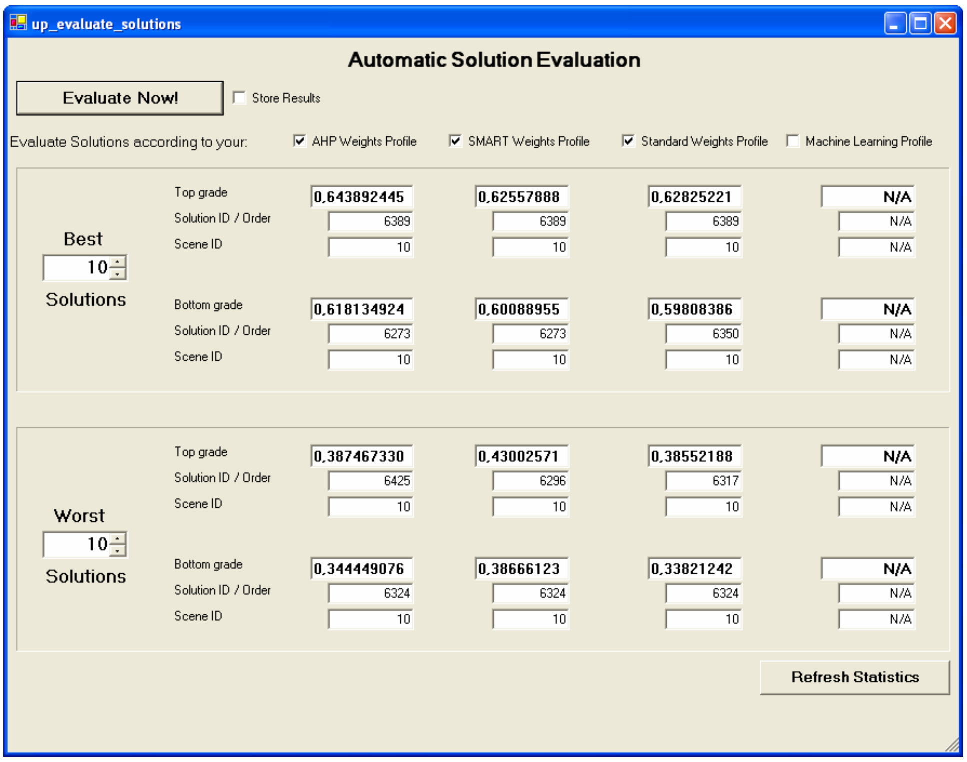
$$b = w \cdot g = \sum_{i=1}^{|A|} w_i g_i$$

where *A* is the set of observed attributes.

	PRA	***PUA***	***NOW***	***NOS***	***NOP***	***PPS***	***SWB***	***Overall Grade***
Middle Values	10	7	1	2	75%	N/A	N/A	
Minimum Values	0	0	0	0	0	0	0	
Maximum Values	25	25	3	5	100%	1	1	
Attribute Values	*11*	*8*	*2*	*1*	*60%*	*1*	*1*	
Attribute Grades	***0.533***	***0.528***	***0.750***	***0.250***	***0.400***	***1.000***	***1.000***	
AHP Weights	0.253	0.304	0.054	0.155	0.056	0.147	0.03	***0.574***
SMART Weights	0.265	0.294	0.088	0.176	0.058	0.088	0.029	***0.547***
RR Weights	0.193	0.386	0.064	0.129	0.077	0.096	0.055	***0.569***

Table 5.3 Overall Solution Grading

The example user input appearing in the previous sections represents a user who has tried to maintain a consistent profile in all three alternative weight assignment methods. It is interesting to see to what degree this is reflected to the weight vectors extracted through each method and the corresponding grade assigned by them to the same solution. In particular, in order to offer a concrete example, we will consider the Solution B appearing in Figure 5.6 and grade it according to the user's profile created by all three weight assignment methods. The corresponding results appear in Table 5.3. The grade of each solution according to each alternative profile appears when the solution is visualised (Figure 5.5) but is calculated before independently of the visualisation as shown in Figure 5.12.



Automatic Solution Evaluation

☐ Store Results

Evaluate Solutions according to your: ☑ AHP Weights Profile ☑ SMART Weights Profile ☑ Standard Weights Profile ☐ Machine Learning Profile

		AHP Weights Profile	SMART Weights Profile	Standard Weights Profile	Machine Learning Profile
Best 10 Solutions	Top grade	0,643892445	0,62557888	0,62825221	N/A
	Solution ID / Order	6389	6389	6389	N/A
	Scene ID	10	10	10	N/A
	Bottom grade	0,618134924	0,60088955	0,59808386	N/A
	Solution ID / Order	6273	6273	6350	N/A
	Scene ID	10	10	10	N/A
Worst 10 Solutions	Top grade	0,387467330	0,43002571	0,38552188	N/A
	Solution ID / Order	6425	6296	6317	N/A
	Scene ID	10	10	10	N/A
	Bottom grade	0,344449076	0,38666123	0,33821242	N/A
	Solution ID / Order	6324	6324	6324	N/A
	Scene ID	10	10	10	N/A

Figure 5.12 Automatic Solution Evaluation Interface – Extreme Grades

In order to further clarify the comparative performance of the alternative weight assignment techniques, Figure 5.13 presents a toy example of two decision support profiles for only two attributes. The grade of a solution for each attribute is the x and y value respectively. The grade of the solution is represented by z. The reason for this simplification is to make the corresponding grade functions representable in the 3D space as surfaces. The horizontal plane in the graph represents a hypothetical grade threshold of 0.7 that could be used to separate the solutions between *approved* and *rejected*.

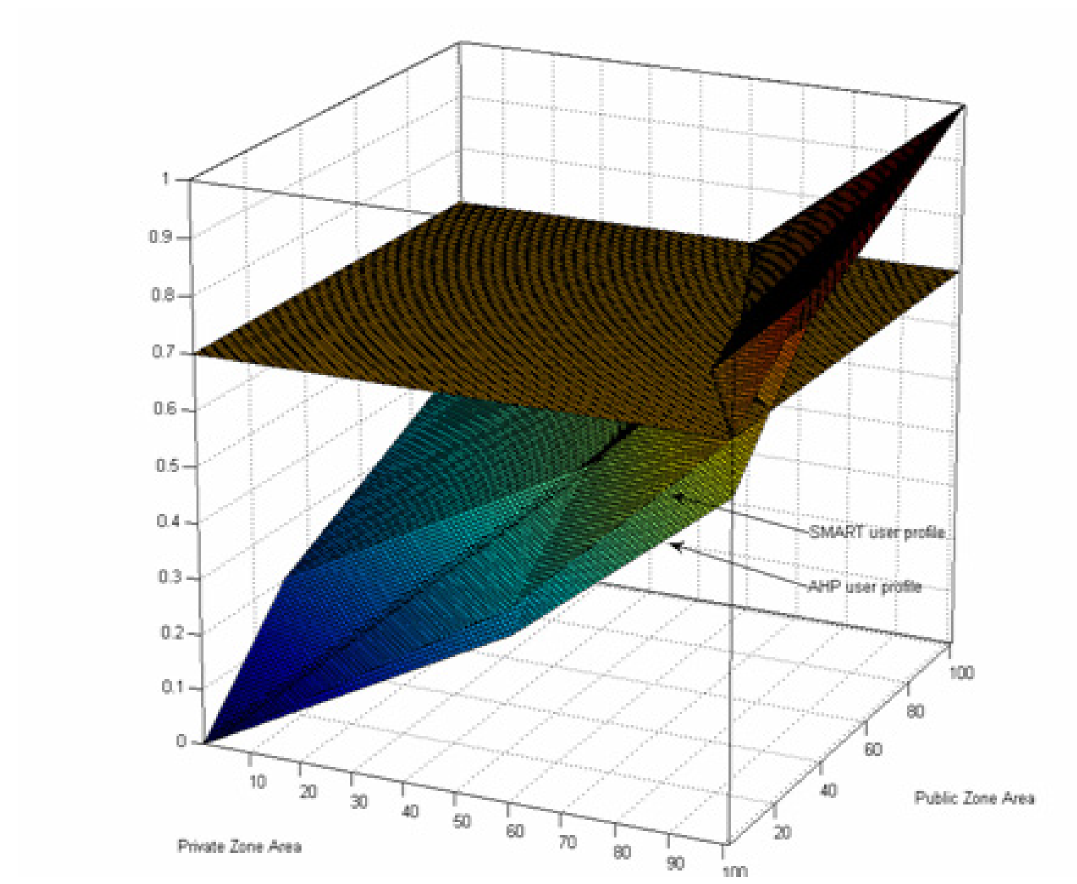


Figure 5.13 Graphical representation of simplified (based only on 2 attributes) alternative Decision Support Profiles

5.6 Machine Learning Component

This is the second component of the Intelligent User Profile Module we have proposed and implemented for the Open-MultiCAD environment. As the name states, it is responsible for adapting the system's response to the user by gradually learning the latter's preferences. It operates concurrently and transparently, during regular system use, automatically evaluating produced solutions while learning from actual user's evaluations. Due to the nature of the environment, we have chosen to apply methodologies of *learning by examples* instead of creating a *set of rules* to reflect user preferences. We have based our approach on the assumption that the average user is not – and probably not willing to be – fully aware of the exact nature of his/her preferences. An additional reason is the fact that, since the average user is not necessarily an architecture expert, it would not be feasible to impose him/her the duty of formulating a rigid set of rules. Last but not least, we have aimed, throughout this work, to an open system with the potential to operate and learn from examples originating from alternative sources with minimal modifications and without the need for prior user engagement.

5.6.1 Methodology

As already discussed in the above, during normal system use, each solution is encoded as a set of values representing its performance with respect to a set of attributes. Note that this evaluation per attribute is influenced, up to an extent, by the user's preferences in the sense that the user has already suggested a custom *middle value* for each attribute, thus defining the value representing 50% performance for the specific attribute. Therefore, each solution is encoded as a vector of normalised grades ranging between 0 and 1. However, overall solution evaluation, i.e. the combined result of these attribute grades remains to be discovered by the Machine Learning Component. During solution visualisation, the user suggests an overall grade for each solution ranging between 0 and 1. The Machine Learning component creates and enhances a Machine Learning profile for each user based on the actual user's evaluation and the corresponding encoded solutions. In particular, the Machine Learning Component operation on the produced solutions can be distinguished as four distinct tasks:

1. Newly generated solutions are automatically evaluated by the Machine Learning Component according to the current state of the Machine Learning Profile. At this point the Machine Learning Component is *unaware* of the actual user's evaluation for the *specific set of solutions.*

2. User evaluated solutions are used by the Machine Learning Component for further refinement of the Machine Learning Profile. The Machine Learning Component *enhances* the user's already existing Machine Learning Profile based on the new set of evaluated solutions. At this point the Machine Learning Component becomes *aware* of the actual user's evaluation for the *specific set of solutions.*

3. The automatic solution evaluation results of Task 1. are compared to the actual user evaluation of Task 2. The current ability of the Machine Learning Component to *generalise* is assessed at this point, i.e. periodically for every new scene, when a new set of user-evaluated solutions has become available.

4. The automatic solution evaluation results are compared to the Decision Support Component results. Assuming that the Decision Support Component offers a minimal consistent representation of the user's preferences, it is checked whether the Machine Learning profile outperforms its Decision Support counterpart.

5.6.2 The Algorithm

We have already discussed the fact that solution generation and evaluation by the user takes place in bursts, each following the submission of a new declarative description. The time lag between two consecutive bursts may vary, depending on the frequency of system usage by the specific user and the complexity of the submitted scene. This fact is an inherent characteristic of the Open-MultiCAD environment architecture. As a result, training data, in regard to the Machine Learning Component, are also available in groups, each originating from the same scene, available at distinct – and probably distant – time points. Hence, for the Machine Learning Component, we had to employ a methodology tailored for *incremental learning,* already discussed earlier in the Machine Learning chapter. The algorithm we have constructed belongs to the family of AdaBoost algorithms [Freund97] and has been based on the Learn++ approach [Polikar01].

The formal definition of the algorithm, in pseudo-code, appears in Algorithm 5.1. Intuitively, the Machine Learning profile consists of a *committee of artificial neural networks*. This committee is built gradually, as a set of *subcommittees*. Each subcommittee is created as soon as a new scene has been submitted and the corresponding solutions have been evaluated by the user. In other words, each subcommittee is responsible for *learning* the user's preferences as expressed in the user's evaluation for a *specific* scene and in relation to the observed attributes.

Each subcommittee contains a number of neural networks not exceeding a pre-defined threshold for maximum subcommittee members. Each one of these neural networks is trained to the point to be able to classify only adequately, i.e. with a relatively high error rate, the solutions produced for the current scene. The high error rate is the reason the learning mechanisms, i.e. the neural networks in the current context, comprising the committee are also called *weak learners* in the relevant literature. The subcommittee is populated with weak learners in a manner that focuses on solutions not correctly classified by previous weak learners of the same subcommittee. Every time a weak learner is added to the subcommittee currently being built, the performance of the committee is checked in order to ensure that the newly added network does not deteriorate overall committee's performance. Overall committee evaluation for any solution is calculated through a *voting procedure* that takes into account the error rate of each weak learner thus recognising higher importance for the votes of those weak learners with reduced error rates. Solution evaluation represents classification of the solution to two or more possible classes. For example, solutions may be classified as *acceptable* or *rejected,* implying two classes, or they may be graded on a scale ranging from 1 to 10 implying ten distinct evaluation classes.

Parameter	***Description***
m	the pre-defined maximum number of members of a subcommittee
p	$p \in \{1..m\}$=the current number of weak learners in the current subcommittee
n	the number of solutions in the current population
i	$i \in \{1..n\}$
k	the number of scenes processed by the Machine Learning Component up to now=the number of committees created up to now
C_k	the current subcommittee
MLC_k	$\{C_1,C_2,\ldots,C_k\}$=the overall committee up to now, including the current composition of the current subcommittee
g_i	grade vector of solution i in the current population
u_i	the user evaluation for solution i
s_i	(g_i,u_i)=sample in the current population=grade vector of solution i coupled with the corresponding user grade.
w_i	sample weight (not connected with attribute weights).
ts	the current training set - created for every weak learner based on weight distribution.
es	current evaluation set – the solutions of the current population not belonging to the ts.
e_p	the product of the weights of all solutions classified incorrectly by the current weak learner
b_p	$=e_p/(1-e_p)$ the *importance* of weak learner's p vote during the overall voting
E	the product of the weights of all solutions classified incorrectly by the current committee

Table 5.4 Machine Learning Component's Algorithm Parameters

```
for each new population of n samples $s_1$, $s_2$, …, $s_n$        // i.e. for each new subcommittee k
    if k>1 then                              // in case the MLC already contains weak learners
        Evaluate new population according to $MLC_{k-1}$
        Update sample weights $w_i$ according to evaluation results
    else
        Initialise sample weights $w_i$ to 1/n
    p=1
    repeat            // for all weak learners to be created in the subcommittee
        repeat        // check new weak learner until overall performance is acceptable
            repeat    // check new weak learner until its performance is acceptable
                **Create ts as a set of samples with at least half the total weight.**
                Create the es with the remaining samples.
                Train a new weak learner using ts.
                Calculate $e_p$ for the current weak learner for ts and es.
            until $e_p \leq 0.5$
            Add current weak learner to the $C_k$
            Evaluate current population according to $MLC_k$
            Calculate E for the current form of the committee
        until $E \leq 0.5$
        Store $b_p=e_p/(1-e_p)$ for the current weak learner
        p=p+1
        Update sample weights according to recent evaluation
        Normalise weights
    **until p>m or E=0**
end of for            // subcommittee k creation loop
```

Algorithm 5.1 Machine Learning Component Algorithm for Incremental Learning

The shaded steps form the major differences from the algorithm proposed in [Polikar01]. In particular,

- Due to the first, we focus on the samples weights, revealing the difficulty in the classification of the specific samples, and not on their number in order to extract the new training subset. In this way, the new weak learner concentrates on the hard-to-classify examples instead of being trained with a lot of easy examples and a few difficult ones in order to reach the half of the total number of available examples. This change has accelerated the generation of adequate weak learners especially in advanced stages of the training process.

- Due to the second, we choose to terminate the addition of new weak learners when the current state of the committee suggests that this is not necessary, i.e. the error has been minimised. In this way the size of the sub-committee is adjusted to the difficulty of the training subset under consideration.

The underlined steps of Algorithm 5.1 above are described in detail in the following. In particular, the *Evaluate population according to MLC* step that takes place at two different points of the algorithm is explained in Algorithm 5.2. The intuition is that each sample is submitted as input to all weak learners. Each weak learner *votes* by suggesting a class for the specific sample. For each class the sum of b's of the corresponding weak learners is maintained. After the sample has been evaluated by all weak learners, the class corresponding to the largest sum of b's, represents the overall decision of the committee for the classification of the specific sample. In other words, the class decided by the overall committee for the specific sample depends not only on the number of votes it receives but also on the importance of these votes.

```
for each sample i in the population
        for each weak learner p in the committee
                Submit the current sample i as input to the current weak learner p
                Add $b_p$ to the sum of the class decided by weak learner p for sample i
        end of for                                                    // weak learners
        Store the class with the largest sum as the overall committee decision for sample i
end of for                                                            // samples
```

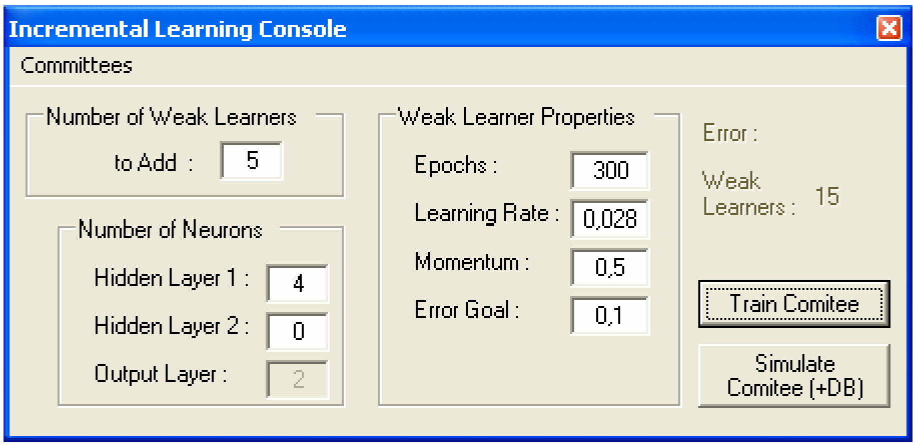
Algorithm 5.2 Algorithm for Samples Evaluation by the Machine Learning Committee

The procedure for sample weights update according to the recent overall evaluation appears in Algorithm 5.3. The intuition in this case is that a sample that has been correctly classified by the overall committee should receive less attention from the weak learners to be created next. In other words, the algorithm concentrates on the hard-to-classify samples, i.e. those erroneously classified by the current form of the committee. This is exploited by the fact that the training set for the next weak learner to be constructed is always created based on the sample weight distribution. In other words, samples with larger weights are more probable to be selected for the next training set thus increasing the probability of the next weak learner to be able to classify them correctly.

```
B=E/(1-E)
for each sample i in the population
        if the sample i is correctly classified by the committee
                $w_i$=$w_i$*B                 // the weight is reduced because B<1 since E<1/2
                                              // else the weight remains the same
end of for                                    // samples
```

Algorithm 5.3 Algorithm for Samples Weights Update

The user interface where the Machine Learning Committee parameters are defined appears in Figure 5.14. A series of experiments is described in the next chapter, covering alternative values of these parameters as well as variations of certain design aspects of the algorithm itself.



Number of Weak Learners to Add : 5

Number of Neurons
Hidden Layer 1 : 4
Hidden Layer 2 : 0
Output Layer : 2

Weak Learner Properties
Epochs : 300
Learning Rate : 0,028
Momentum : 0,5
Error Goal : 0,1

Error :
Weak Learners : 15

Figure 5.14 Machine Learning Component Console

5.7 User Profile Database

All information regarding the Intelligent User Profile Module operation and the relevant user data are stored in a dedicated database. The Entity Relationship schema of this database appears in Figure 5.15.

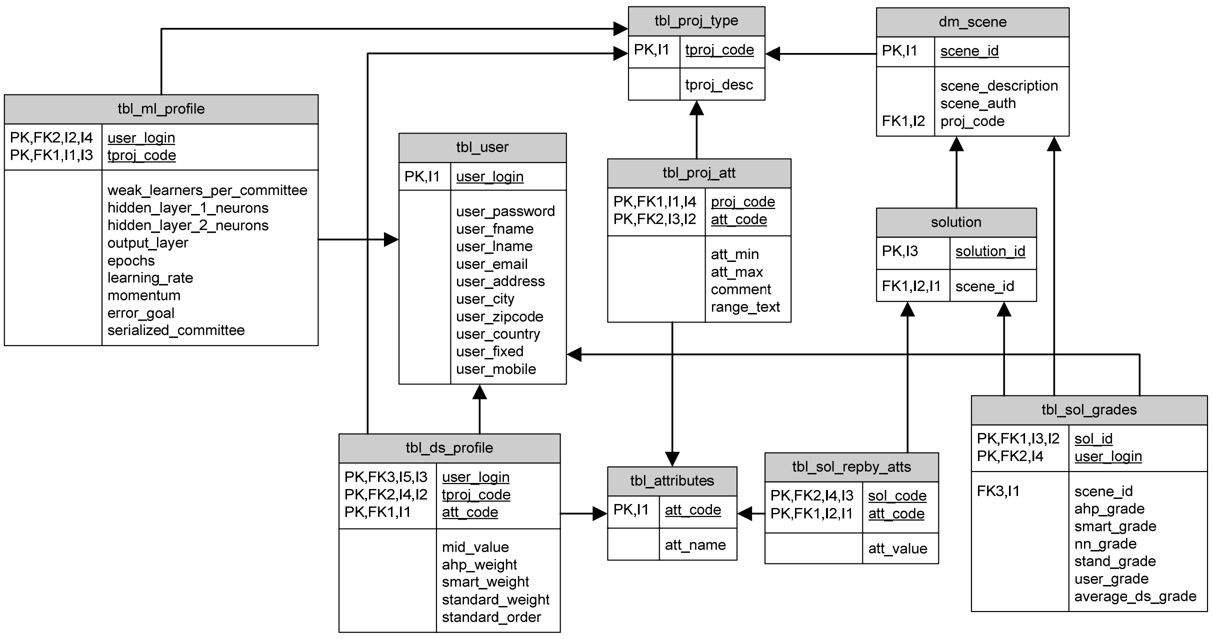


Figure 5.15 Entity Relationship Schema of the User Profile Database

The entities *dm_scene* and *solution* contain information regarding each submitted scene and the connection between a solution's unique identification and the scene for which it has been generated. Both entities already existed in the MultiCAD database, discussed in the Declarative Modelling Chapter and have been included in Figure 5.15 in order to clarify the connection between the two databases. Notice that no other reference exists to entities of the MultiCAD database although information regarding the geometric representation of solutions is used for the calculation of the attribute values stored in the User Profile Database.

The latter contains all Decision Support Profile information connected to a specific user. This information includes the weights from all alternative methods used for weight assignment coupled with the middle values selected by the specific user for each observed attribute. Notice that each project type, i.e. residence, office, hotel, etc., is related with a different set of observed attributes. Therefore, the Decision Support profile contains not only the user's identification but also the identification of the corresponding project type.

The situation is similar with respect to the Machine Learning Profile information. However, in this case, more information has to be stored regarding not only the configuration characteristics of the committee, for example *number of weak learners per subcommittee*, or also general characteristics common for all neural networks participating in the committee, for example *learning rate* and *momentum,* but also the specific characteristics of each neural network. In particular, common design information regarding the neural networks is stored in the database itself as shown in the entity *tbl_ml_profile* of the Entity Relationship schema. However, the *current state* of all neural networks participating in the committee is stored as a file whose name is stored in the *serialised_committee* field of the aforementioned entity. The reason for this separation of data is mainly practical in the sense that, due to technical reasons of the implementation, it was simpler to export the current state of the overall committee as a serialised object instead of storing every detail of it in the database and subsequently restoring it in order to resume training. Nevertheless, complete information about the state of each neural network is available within the User Profile Module in case it is needed for additional insight.

Observed attributes are stored as a separate entity and are connected to project types according to the needs of the latter. Generally, during normal use of the system on behalf of the user this information is not modifiable by the user. However, the administrator is fully capable of modifying the connection between attributes and project types by simple modification of the contents of the corresponding *tbl_proj_att* entity. This entity also contains information regarding the characteristics of the specific observed attribute in the context of the specific project type. For example, the maximum and minimum values allowed are different for the same attribute, e.g. *number of bedrooms*, in the context of different project types, e.g. *residence* or *hotel.*

5.8 Performance Indices

In order to be able to compare the two components we provide the means to measure their performance. We concentrate on the automatic solution evaluation stage that takes place after solution generation and solution evaluation by the user. In particular, in the following discussion, solution generation has already taken place and the methods have been applied to the results. Moreover, for the sake of simplicity, only two possible classes for solution evaluation are considered. Hence, the application of each method to the solutions yields the corresponding subset of approved solutions. Finally, we mention only two methods in the following whereas, in practice, more alternatives – due to alternative weight assignment, alternative network adjustments, etc. – may be concurrently evaluated. The methodology described below is trivially extended to support the comparison of more than two methods.

In particular, let us define the following sets:

G = The set of solutions generated based on the specific description of a scene and evaluated by the user.

U = The set of solutions approved by the user, i.e. solutions in G that comply with the user preference. Formally, $U \subseteq G$. In the following we refer to the members of U as approved solutions implying user approval.

G–U = The set of discarded solutions, i.e. the generated solutions that are rejected by the user.

M_1 = The set of solutions in G approved by Method 1. Formally, $M_1 \subseteq G$.

M_2 = The set of solutions in G approved by Method 2. Formally, $M_2 \subseteq G$.

|S| = The number of members of any set S.

Based on the above we have:

M–U = The set of solutions erroneously approved by M since they belong to M but not to U.

U–M = The set of solutions erroneously discarded by M since they belong to U but not to M.

We may now define the overall *error* for a method M based on the solutions erroneously evaluated by it. In particular:

$$E = \frac{|M - U| + |U - M|}{|G|}$$

Moreover, we may define the *hit rate* of each method as:

$$HR_i = \frac{|M_i \cap U|}{|U|}, \quad i \in \{1,2\}$$

i.e. the percentage of approved solutions captured by the specific method.

The ratio of approved vs. total solutions selected by each method can also be used as measurement of their performance:

$$PR_i = \frac{|M_i \cap U|}{|M_i|}, \quad i \in \{1,2\}$$

There are more than one ways to define a miss rate. We define it as the percentage of discarded solutions that are selected by the method, expressed as:

$$MR_i = \frac{|M_i - U|}{|G - U|}, \quad i \in \{1,2\}$$

However, it may be the case that only a small number of the generated solutions fulfil the user preference. This is mainly due to time limitations posed by the requirement for human visual inspection. On the other hand, |G| greatly depends on the description and can vary significantly. Therefore, we need to relate the size of the error for each method with |U| instead of a quantity including |G|. Hence, we alternatively define:

$$MMR_i = \frac{|M_i - U|}{|U|}, \quad i \in \{1,2\}$$

and interpret a lower value as a better performance. Intuitively, this interpretation implies that a method should not select many discarded solutions when only a few preferred solutions exist. For example, when this rate is more than 1 the method gives more discarded solutions than the total number of approved solutions. Instead of 1, another value may be selected to reflect a specific performance threshold.

The above are clarified in the example Venn diagram of Figure 5.16, representing a general case (i.e. no intersection is empty, no two sets are equal). For the sake of simplicity of the picture, the total number of solutions is rather small, i.e. |G| is only 35 whereas, generally, this may not be the case. Nevertheless, for the specific example, we have the following numbers:

$|G| = 35$, $|U| = 10$, $|M_1| = 6$, $|M_2| = 10$, $|U - M_1|=7$, $|U - M_2|=6$

$|M_1 \cap U| = 3$, $|M_2 \cap U| = 4$, $|M_1 - U| = 3$, $|M_2 - U| = 6$.

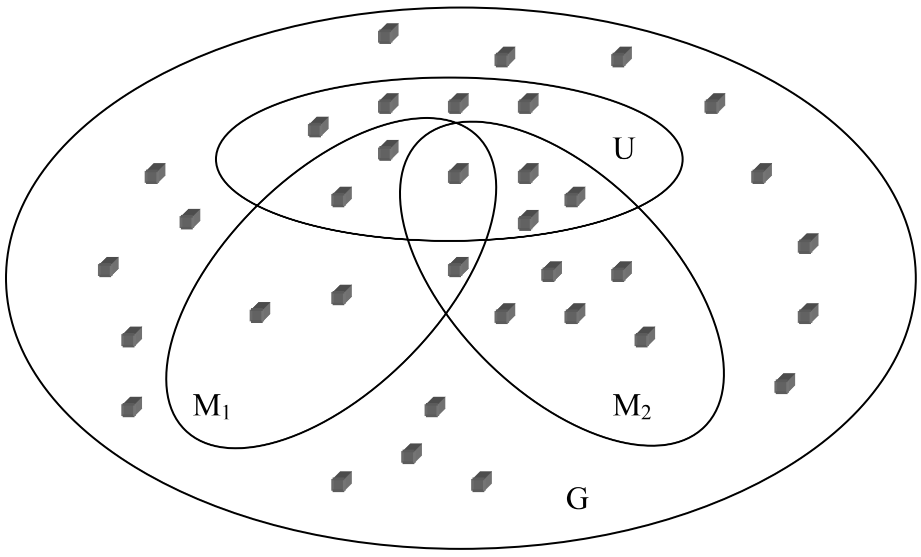


Figure 5.16 Example Methods Performance

Therefore for M_1: E = 10/35, HR_1 = 3/10, MR_1 = 3/25, MMR_1 = 3/10, PR_1 = 3/6 and for M_2 we have: E = 12/35, HR_2 = 4/10, MR_2 = 6/25, MMR_2 = 6/10, PR_2 = 4/10

Method	***Error***	***Hit Rate***	***Performance Ratio***	***Miss Rate***	***Modified Miss Rate***
Method 1	10/35 = 28.6%	3/10 = 30%	3/6 = 50%	3/25 = 12%	3/10 = 0.3
Method 2	12/35 = 34.3%	4/10 = 40%	4/10 = 40%	6/25 = 24%	6/10 = 0.6
Extreme Case 1	9/35 = 25.7%	1/10 = 10%	1/1 = 100%	0/25 = 0%	0/10 = 0.0
Extreme Case 2	25/35 = 71.2%	10/10 = 100%	10/35 = 28.6%	25/25 = 100%	25/10 = 2.5

Table 5.5 Performance Indices for Example Methods of Figure 5.16

M_2 could be considered a worse (because of the higher error) or a better (because of the higher hit rate) method than M_1 depending on the interpretation of these numbers. Ideally, error should be equal to 0%, hit rate should be equal to 1, miss rate equal to 0 and performance ratio equal to 100%. Notice, however, that a method selecting only one preferred solution every time it is invoked (Extreme Case 1) would yield a performance ratio of 100% and a low error, depending on the total number of approved solutions, without necessarily representing an optimal method as shown by the low hit rate. On the other hand, simply selecting all produced solutions (Extreme Case 2) maximizes the hit rate but yields a low performance ratio. In any case, Extreme Case 1 appears to represent a method with acceptable performance whereas Extreme Case 2 represents a trivial approach of no practical use. Therefore, a high Performance Ratio appears to be a necessary, although not sufficient, indication of an efficient method and it becomes apparent that we need a combination of these indices in order to evaluate in greater detail the performance of alternative configurations.

6 Experiments

The prototype described in the previous chapter has undergone a series of experiments in order to assess the degree of its ability to capture and maintain a reliable model of user preferences. As it is usually the case with complicated systems, it is not only the alternative inputs that lead to different experimental results. Several parameters of the system may be assigned alternative values, within their valid range, thus yielding alternative versions of the system itself. Typically, each one of these versions may respond differently to the submitted inputs. In particular, for the Intelligent User Profile Module, it may be the case that it exhibits improved performance for certain settings for the solutions obtained from a specific scene but the same settings may not perform equally well for the solutions corresponding to an alternative scene. Alternatively, it may be the case that the best performing settings vary depending on the input.

6.1 Methodology

In order to achieve a consistent set of results reflecting the capabilities of the implemented module, we have organised the experiments in three distinct stages following a pre-processing stage.

1. **User Representative Scenes (Pre-processing).** Before the actual experiments could take place we had to create a set of scenes that would be submitted to the system in order to obtain the corresponding sets of solutions. These scenes had to provide adequate flexibility and reduced size of the solution population. The former is needed in order to produce diverse solutions of varied performance. The latter is also necessary in order to allow the repetitive use of the complete solution population to

assess alternative settings within acceptable time. The result of this pre-processing stage is a set of representative scenes, each accompanied by the solutions produced for it.

2. **Initial Settings Evaluation.** During this stage we have used the set of representative scenes in order to evaluate alternative settings for the parameters of the system. Some of the parameters of the mechanisms employed have received increased focus during this stage, since they are directly connected with the quality of the input produced by Open-MultiCAD itself. The outcome of this stage is the set of top performing alternative module settings to be used during the next stage.
3. **Module Evaluation.** This stage forms the main core of the experiments, focusing on the response of the system with respect to alternative inputs representing different kinds of users and user behaviours. Moreover, because of the nature of the Open-MultiCAD solution generator, alternative parameters regarding the quality of the input data have also been taken into account with respect to each user. The set of representative scenes defined during the first stage is extended to fulfil this requirement. The result of this stage is an overall module evaluation with respect to the alternative settings.
4. **Fine Tuning.** After the system performance has been assessed based on the alternative settings, the best performing of the latter are subject to a final series of experiments in order to further improve their performance. The stability of the overall results is also examined during this stage.

User's evaluation for each solution plays a crucial role to the experiments as well as to the overall system evaluation and parameterisation. In particular, the evaluation method could range from a binary classification to a real number grading scheme. We have chosen to create an open module, able to cope with complex grading schemes with only minor modifications of the prototype. However, we have started and eventually concentrated, for the series of experiments described in the following, on the binary classification method. The main reason for this has been the already large number of parameters that could affect the behaviour of the prototype. In other words, we have tried to focus on system settings optimisation and evaluate the algorithms used with respect to the minimal example of human solution evaluation before trying to adapt the system to more complex grading schemes. Nevertheless, all modules comprising the Intelligent User Profile Module are parametric and capable to support a practically limitless number of alternative evaluation classes instead of the minimal two.

6.2 User Representative Scenes

At the first stage of this series of experiments we have created a set of alternative declarative descriptions as prototype examples. These scenes have been constructed in a way that allows the participation of all observed attributes through a wide range of attribute values. The main goal has been to offer a broad solution space that would make interesting not only the comparison of solutions among the ones generated for the same declarative description but also among solutions generated by different scenes. Last but not least, the scenes have been aimed to represent a set of realistic habitation buildings. Throughout these experiments we have focused on habitation project types in order to be able to concentrate on the tuning of the module upon a common foundation of comparable examples.

It is also important to notice, at this point, that during the subsequent stages of initial settings and module evaluation, each scene used in an experiment is always filtered through the specific – artificial or actual – user's profile. This implies that a solution set may receive an entirely different evaluation when applied to alternative user profiles. Therefore, as it has often been the case during the experiments, a solution which is approved according to a specific user's profile may be rejected in regard to another user.

6.2.1 The Representative Scenes

In the following we present the declarative description of the scenes used for the experiments and a pair of example solutions for each scene. In order to enhance readability, we employ the same tree structure diagram that was used for the same purpose in the Declarative Modelling chapter. The names we have used for the scenes are intuitive, *not* descriptive of any particular scene properties or architectural style, just in order to facilitate reading of the subsequent experiments. We have used alternative dimensions and shapes for the site during the solution generation maintaining, however, similar site sizes. Moreover, since the observed attributes are all two-dimensional, we have set a fixed height for all objects of the representative scenes. Nevertheless, our purpose has been to construct a wide range of solutions that could be used for the assessment of the alternative Intelligent User Profile Module parameter values.

6.2.1.1 Naxos House

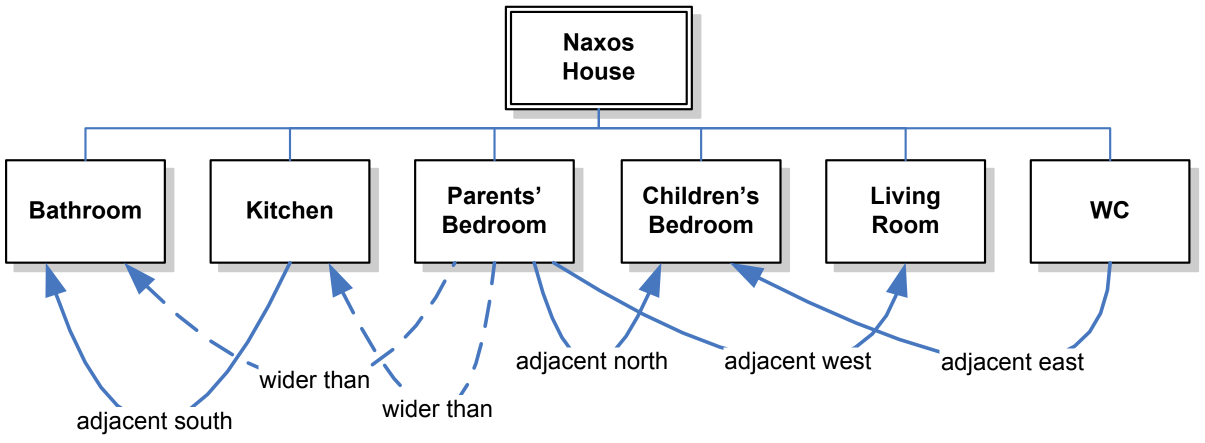


Figure 6.1 First representative scene – site area for solution generation: 6×4 units

It is interesting to see how the declarative design process may lead to solutions with unexpected properties, a fact that is exactly its main advantage: based on an abstract description, it leads to a wide range of alternative interpretations. Figure 6.2(a) shows one of the best performing solutions – according to a specific user's profile – generated for the declarative description of Figure 6.1. Notice how the proportional size of the WC room (yellow) at the bottom right corner, i.e. south-east, is somehow unnatural when compared with the other rooms. This is due to the fact that there is no explicit declarative relation or property regarding this quantity therefore the size of the corresponding room is restricted only by the site dimensions and the positioning of the other rooms. On the contrary, the bathroom (light blue) on the top left, i.e. north-west, has been restricted by the declarative description to be narrower than the parents' bedroom. What may seem as unnatural or odd in this case could be an alternative idea for the designer in another case.

Notice also how the nature of the observed attributes affects the overall solution performance. The specific sample has a clear separation of the public and private zones – the former consisting of the WC (yellow) and the Living Room (green) – and contains no oblong rooms. Last but not least, it features an increased total area of private spaces which, according to the specific user's decision support profile, was an important attribute. All these observations do not hold for the sample appearing in Figure 6.2(b) and this fact is reflected in the solution evaluation. We will discuss these observations in detail in the following sections covering the experiments.

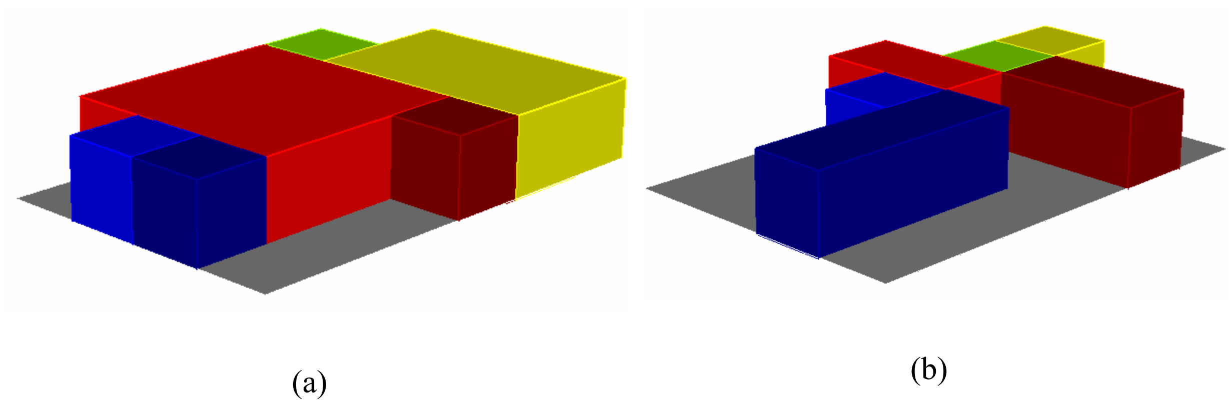


Figure 6.2 One of the best (a) and worst (b) performing solutions for the Naxos scene declarative description

6.2.1.2 Rentis House

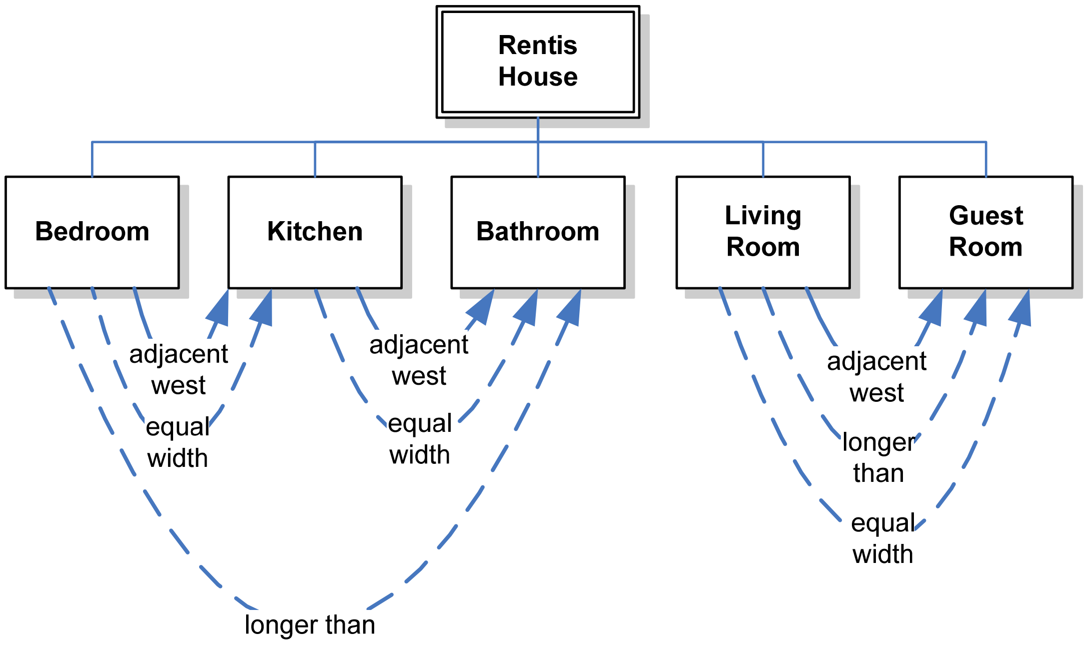


Figure 6.3 Second representative scene – site area for solution generation: 5×5 units

In this example we have avoided to connect declarative objects of the private area with those of the public area thus allowing all possible overlaps. Moreover, we have connected certain properties in such a way that allows multiple alternative placements of groups of objects in order to achieve a wide range of values for the observed properties. Example solutions of the declarative description of Figure 6.3 appear in Figure 6.4. Notice how the public zone objects – living room (yellow) and guest room (green) – have been placed on the north side of the site in (a) and on the south side of the site in (b) since there is no relation connecting objects of the two zones.

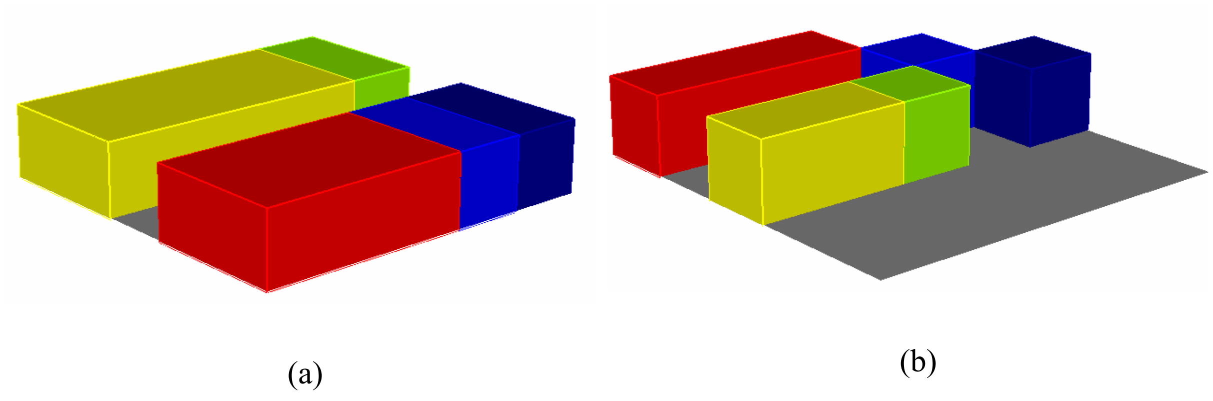


Figure 6.4 One of the best (a) and worst (b) performing solutions for the Rentis scene declarative description

6.2.1.3 Piraeus House

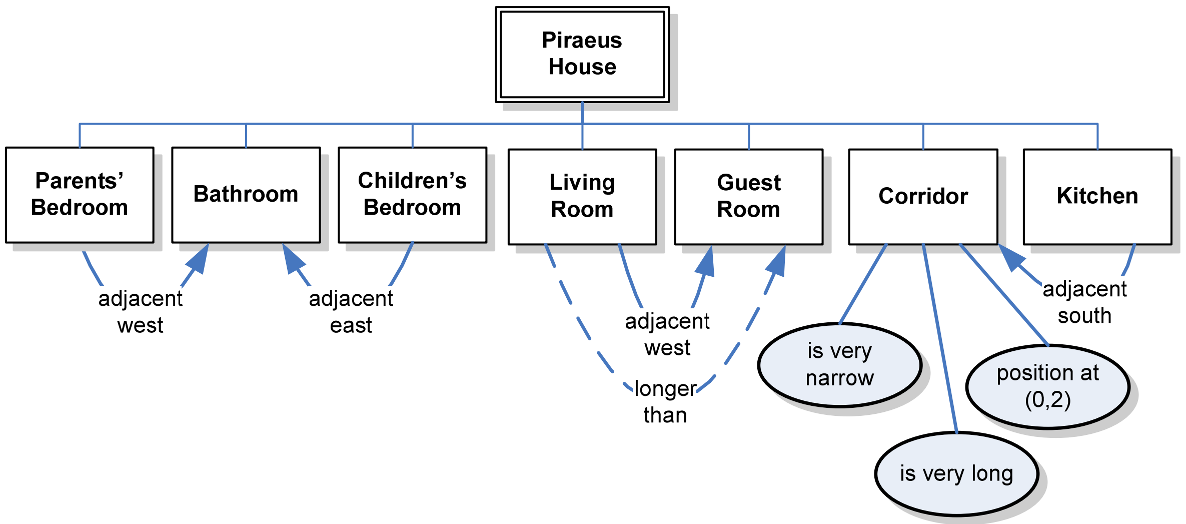


Figure 6.5 Third representative scene – site area for solution generation: 5×5

In this scene we have included a corridor object of fixed position and very restricted dimensions in order to reduce the solution space. In particular we have allowed all objects to be positioned either on the north or the south side of the corridor except for the bathroom.

Therefore, zone separation has depended on the positioning of the other objects with respect to the corridor. Moreover, although the corridor is oblong by construction, the intelligent user profile module *does not* consider corridor objects in the calculation of the *non-oblong rooms percentage* attribute, as already discussed in the previous chapter. Hence, the overall solution performance is not affected by the presence of the corridor while the solution space is reduced in a uniform manner, with respect to the observed attributes. Regarding the examples appearing in Figure 6.6, the performance of solution of (b) is mainly affected by the small size of the objects and the overlap of public spaces – living room (yellow) and guest room (green) – and private spaces. Also notice the presence of a south-western bedroom (red) in (a) that gives a *true* value to the corresponding binary observed attribute.

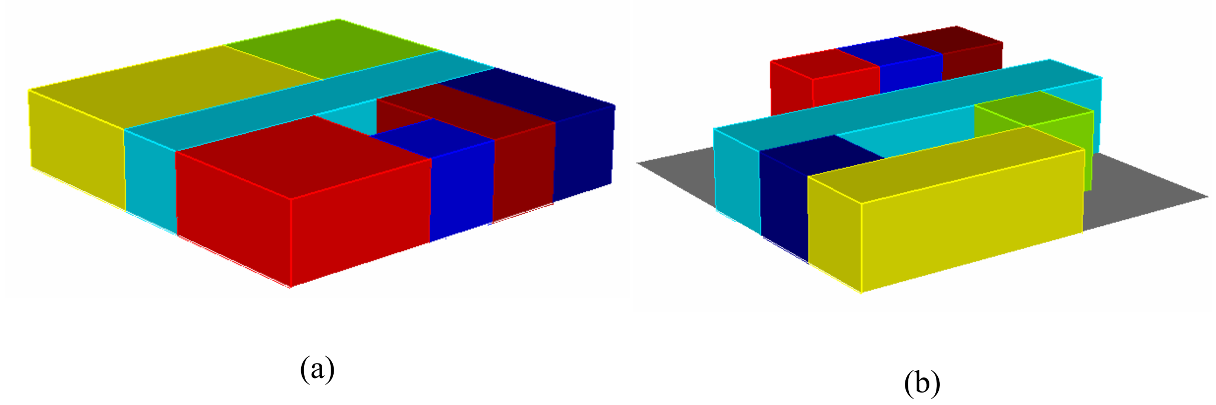


Figure 6.6 One of the best (a) and worst (b) performing solutions generated for the Piraeus declarative scene description

6.2.1.4 *Kalamaki House*

The description of Figure 6.7 represents a simple scene with minimal relations and a declarative object that does not directly participate in the declarative description. However, this object spans across the entire width of the site, thus forcing the remaining objects towards one of its sides due to their relations. Space overlapping is avoided only in extreme cases of interpretation of the *adjacent* relation by the solver where the participating objects are only marginally in touch. An example of such a case appears in Figure 6.8(a) where the guest room (green) is adjacent south to the bedroom (red) as well as the kitchen (brown) is adjacent south to the bathroom (blue), both in a marginal manner, having only an edge in common with the adjacent object.

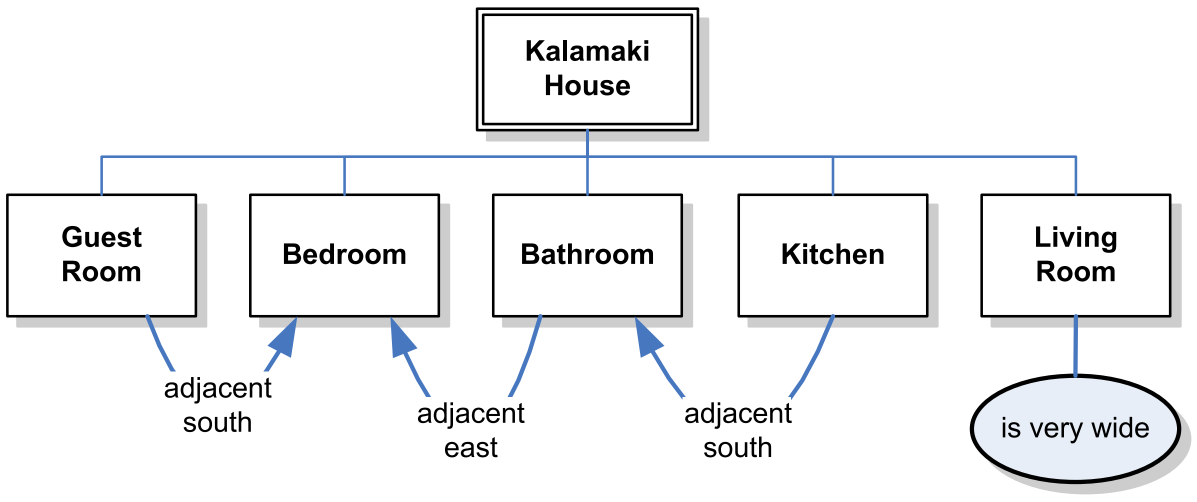


Figure 6.7 Fourth representative scene – site area for solution generation: 6×4 units

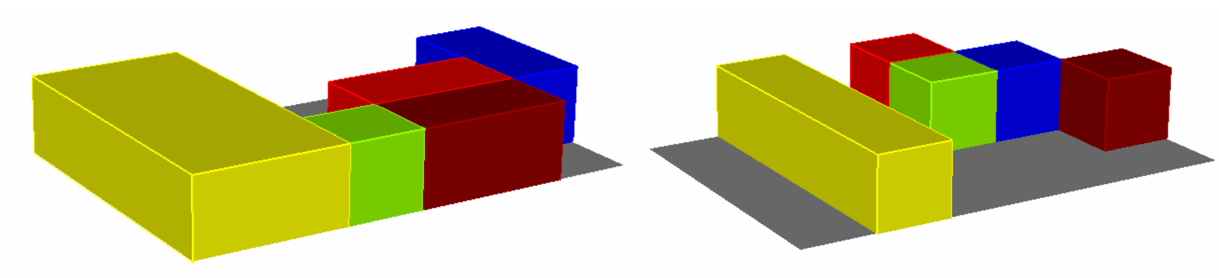
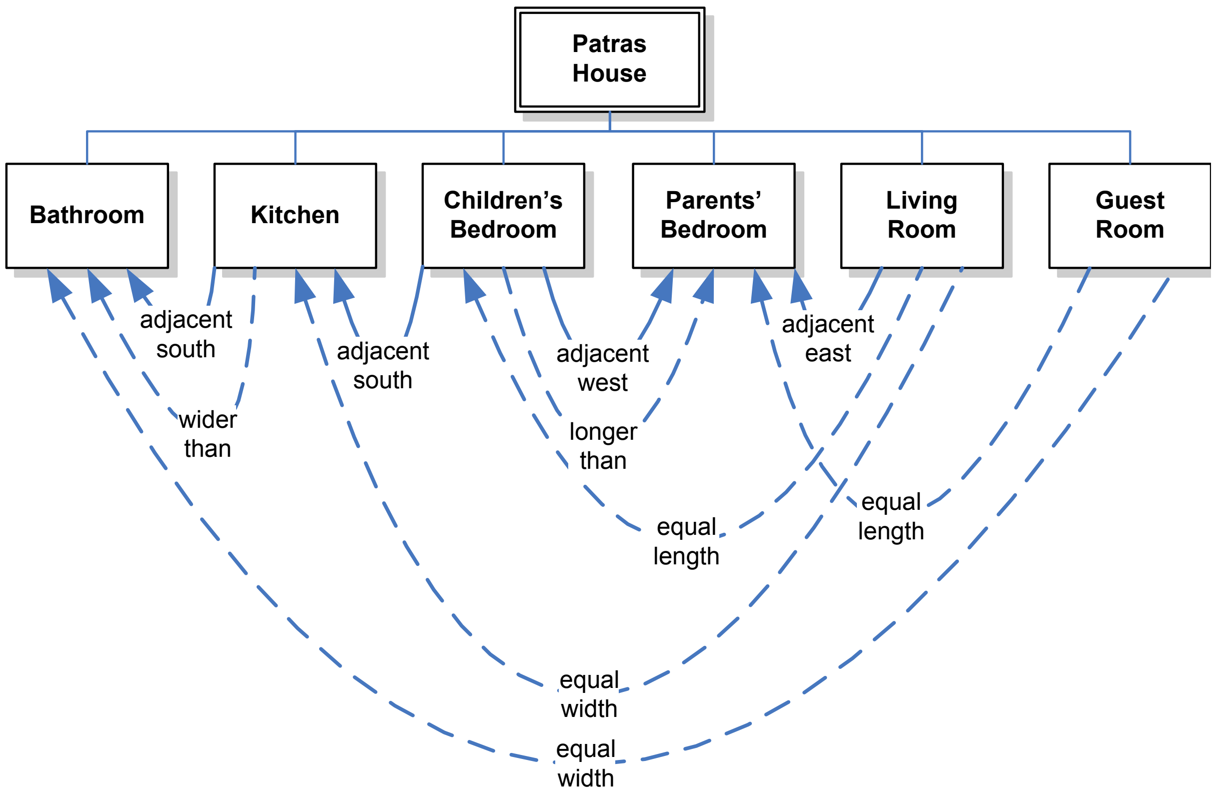
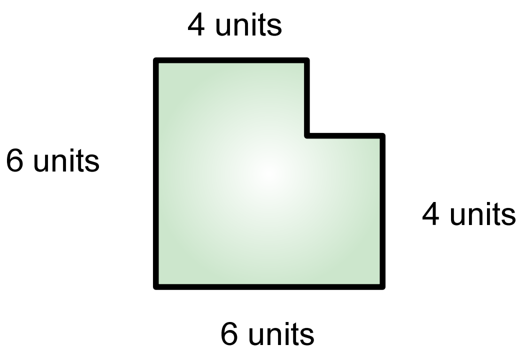


Figure 6.8 One of the best (a) and worst (b) performing solutions generated for the Kalamaki declarative scene description

6.2.1.5 Patras House



(a)



(b)

Figure 6.9 (a) Fifth representative scene (b) Site area for solution generation: L-shaped 6×6 units, thickness 4 units

For this scene we have focused on correlational relations, largely connecting morphological properties of the declarative objects participating in the scene description. Moreover, we have requested an L-shaped site for solution generation in order to explore arrangements in a contour other than the traditional rectangle. Figure 6.10 presents two of the generated solutions for the specific declarative description. Notice how the guest-room (green) is freely positioned due to the lack of spatial relations referring to it while maintaining the same width with the bathroom (dark blue) and the same length with the parents' bedroom (brown).

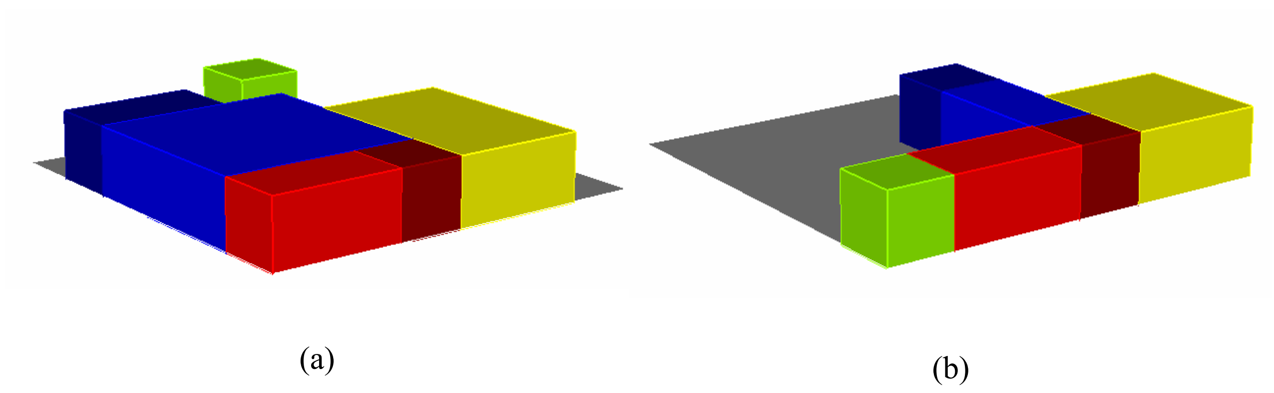


Figure 6.10 One of the best (a) and worst (b) performing solutions generated for the Patras declarative scene description

Since the aforementioned scenes produce large solution populations we have used only a subset for the experiments as shown in Table 6.1. The main reason for this has been the fact that, during actual system use, the average user will be able to evaluate at most a few hundreds solutions. Therefore, training the Machine Learning Component with hundreds of thousands of automatically produced and artificially evaluated training samples would offer better insight regarding the mechanism itself but in an completely unrealistic context. Hence, we have applied automated evaluation on a subset of the produced solutions and submitted these solution sets as training sets to the Machine Learning Module. The automatic evaluation, i.e. the artificial user's feedback, has been obtained based on the Decision Support Profile of actual users and the corresponding solution grades implied by it. The advantage of this approach is that it is based on realistic data regarding the importance of the observed attributes for the specific user while producing evaluations that are consistent throughout alternative scenes. In a sense, we have focused the evaluation of the Machine Learning Component on capturing the preferences of users that would remain consistent to their preferences throughout the use of the system.

Declarative Scene Description	***Overall Solution Population***	***Evaluated Solutions***
Naxos House	670260	670
Rentis House	21184	411
Piraeus House	306255	621
Kalamaki House	212096	530
Patras House	511450	511

Table 6.1 The solutions produced for the representative scenes and the subsets used for the Machine Learning Component initial settings evaluation

6.3 Initial Settings Evaluation

The next stage of the experiments comprised the evaluation of the module against alternative settings for the parameters controlling the Machine Learning Component. These parameters are summarised in Table 6.2. Most of them refer to the generic neural network used to represent the *weak learner* whereas the number of members per scene refers to the *sub-committee* formed by these weak learners every time a new scene is submitted for training. The Typical Values column represents the best performing configuration that was concluded after numerous experiments that took place during the module's development, debugging and testing. This is the reason we use them as basis in the following experiments, varying one or more configuration parameters at a time and observing the effect in performance.

In the following we present the evolution of the Machine Learning Component with respect to the representative scenes for alternative system settings. In particular, the MLC has been trained incrementally, one scene at a time, thus simulating regular system use. For these preliminary experiments, we have used an artificial user evaluation based on the evaluation yielded by the Decision Support Component; the details of the corresponding Decision Support Profile appear in the next section, discussing the main Module Evaluation. For this reason, we do not compare the performance of the two components at this stage and we concentrate on the performance of the MLC with respect to the expected solution classification.

Parameter	***Typical Value***	***Alternative Value(s) Examined***			***Parameter Reference***
weak learners added per scene (WL)	5	10	3	1	committee
first hidden layer neurons (HL1)	4	2			weak learner
second hidden layer neurons (HL2)	0	2			weak learner
epochs (E)	1000	300			weak learner
learning rate (LR)	0.028	0.01	0.09	0.15	weak learner
momentum (M)	0.5	0.1	0.3	0.9	weak learner
error goal (EG)	0.1	0.3			weak learner

Table 6.2 Parameters and alternative values used during initial settings evaluation

Each table in the following shows the erroneously classified solutions by the MLC after each scene has been used for training. In particular, each column in the table represents a new scene that has been submitted and subsequently evaluated by the MLC whereas each row shows the error for the particular scene throughout the training. The shaded boxes represent the Error, as defined in the previous chapter, for scenes the MLC has already been trained with and the boldface percentage represents the Error for the most recent scene used for incremental training.

6.3.1 Initial Settings Experimental Results

In the following we present the explicit results of the initial settings experiments accompanied with an interpretation of the system’s behaviour regarding the specific parameter values. We commence this presentation with one of the best performing combinations and subsequently compare it with alternative configurations.

6.3.1.1 Configuration 1

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House	
Naxos House	**1.94%**	11.04%	6.57%	9.55%	3.73%	
Rentis House	21.90%	**4.38%**	13.38%	9.49%	11.68%	
Pireaus House	28.99%	12.56%	**1.45%**	15.14%	14.49%	
Kalamaki House	41.89%	2.26%	10.38%	**0.94%**	2.08%	
Patras House	10.76%	17.42%	0.59%	17.42%	**3.33%**	
Average Training Error	1.94%	7.71%	7.13%	8.78%	7.06%	6.53%
Average Generalisation Error	25.88%	10.75%	5.48%	17.42%		14.88%
Overall Error	21.09%	9.53%	6.47%	10.51%	7.06%	10.93%

Table 6.3 WL=5, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

In this experiment the module seems to improve generalisation results after each scene. However, when the *Patras House* remains as the only unseen example description the generalisation error increases revealing potential particularities of the specific scene and dense placement of alternative classes examples in the solution space. It is important to notice that, despite the fact of the incremental nature of the training, the training error generally decreases or remains in similar levels after the first scene and for each subsequent description submitted, mainly due to the large number of epochs allowing the new members of the committee to closely capture the new scene. The overall error is more than satisfactory for this style of learning whereas the generalisation error seems promising. This configuration exhibited the best overall performance and the smallest training error whereas it ranked second only in generalisation ability.

6.3.1.2 Configuration 2

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	300	0.15	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**7.16%**	10.30%	4.33%	10.90%	9.85%
Rentis House	22.14%	**8.52%**	23.36%	20.92%	25.30%
Pireaus House	31.24%	16.10%	**12.56%**	13.53%	9.82%
Kalamaki House	25.47%	3.21%	4.53%	**1.89%**	4.72%
Patras House	26.03%	26.61%	6.07%	20.35%	**9.00%**

Average Training Error	7.16%	9.41%	13.42%	11.81%	11.74%	10.71%
Average Generalisation Error	26.22%	15.31%	5.30%	20.35%		16.79%
Overall Error	22.41%	12.95%	10.17%	13.52%	11.74%	14.16%

Table 6.4 WL=5, HL1=4, HL2=0, E=300, LR=0.15, M=0.5, EG=0.1

In this experiment we have reduced the number of epochs substituting for them – in principle – with a more aggressive learning rate. In practice, the combination's performance is inferior both in terms of training and generalisation error.

6.3.1.3 Configuration 3

Weak Learners	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*
5	2	2	1000	0.028	0.5	0.1

Scene ID	Error after training with solutions of scene				
	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**2.99%**	14.33%	3.13%	7.01%	10.30%
Rentis House	25.79%	**6.57%**	24.57%	20.19%	20.19%
Pireaus House	31.88%	15.14%	**15.62%**	12.08%	15.94%
Kalamaki House	43.58%	6.42%	21.13%	**4.15%**	4.15%
Patras House	12.52%	25.83%	12.33%	5.68%	**11.55%**

Average Training Error	2.99%	10.45%	14.44%	10.86%	12.43%	10.23%
Average Generalisation Error	28.45%	15.79%	16.73%	5.68%		16.66%
Overall Error	23.35%	13.66%	15.36%	9.82%	12.43%	14.92%

Table 6.5 WL=5, HL1=2, HL2=2, E=1000, LR=0.028, M=0.5, EG=0.1

In this case we have enhanced each weak learner to comprise 2 hidden layers however the configuration does not exhibit better performance when compared to the same settings with the equal total number of hidden neurons.

6.3.1.4 *Configuration 4*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
1	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**4.78%**	4.78%	6.12%	11.94%	6.12%
Rentis House	20.44%	**20.44%**	17.03%	15.57%	17.03%
Pireaus House	13.04%	13.04%	**5.96%**	19.00%	5.96%
Kalamaki House	21.70%	21.70%	9.62%	**3.21%**	8.68%
Patras House	4.31%	4.31%	4.70%	37.18%	**4.70%**

Average Training Error	4.78%	12.61%	9.70%	12.43%	8.50%	9.60%
Average Generalisation Error	14.87%	13.02%	7.16%	37.18%		18.06%
Overall Error	12.85%	12.85%	8.69%	17.38%	8.50%	12.05%

Table 6.6 WL=1, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

This experiment stemmed from the idea that the artificial user preferences might not need many weak learners to be captured but, instead, only one for each scene. In practice, the configuration performed worse than the multi-member committees revealing that the dense placement of solutions belonging to different classes makes it difficult for the mechanism to locate the boundaries using just one weak learner per scene. It is also characteristic for this configuration the fact that the second scene does not alter the component's behaviour. This is due to the fact that the weak learner corresponding to the first scene features high *voting weight*, due to its increased performance, thus out-weighing the second weak learner in every voting. Only after a third weak learner is added to the committee the results start to vary. Nevertheless, this configuration was positioned above average in terms of overall performance.

6.3.1.5 Configuration 5

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	300	0.028	0.9	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**18.06%**	14.03%	10.45%	12.39%	14.78%
Rentis House	33.09%	**6.08%**	20.19%	25.79%	26.03%
Pireaus House	34.78%	16.59%	**7.73%**	14.98%	20.93%
Kalamaki House	64.91%	7.36%	4.91%	**4.72%**	2.45%
Patras House	10.18%	49.12%	36.40%	14.48%	**7.83%**

Average Training Error	18.06%	10.06%	12.79%	14.47%	14.40%	13.96%
Average Generalisation Error	35.74%	24.35%	20.65%	14.48%		23.81%
Overall Error	32.20%	18.64%	15.94%	14.47%	14.40%	19.13%

Table 6.7 WL=5, HL1=4, HL2=0, E=300, LR=0.028, M=0.9, EG=0.1

This configuration represents an effort to emphasise on the *weakness* of the committee members and rely more on the collective decisions of the overall committee. Practically, the configuration exhibits decreased performance both in terms of generalisation and training error.

6.3.1.6 Configuration 6

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.09	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**2.54%**	11.49%	3.13%	9.70%	5.07%
Rentis House	23.11%	**2.43%**	22.14%	21.65%	15.09%
Pireaus House	18.68%	2.58%	**11.27%**	12.88%	13.20%
Kalamaki House	29.81%	10.57%	15.85%	**1.51%**	7.92%
Patras House	11.35%	26.03%	7.83%	7.63%	**4.11%**

Average Training Error	2.54%	6.96%	12.18%	11.44%	9.08%	8.44%
Average Generalisation Error	20.74%	13.06%	11.84%	7.63%		13.32%
Overall Error	17.10%	10.62%	12.04%	10.68%	9.08%	11.90%

Table 6.8 WL=5, HL1=4, HL2=0, E=300, LR=0.09, M=0.5, EG=0.1

In this case, we modified the most successful configuration by increasing the learning rate. The results demonstrated a decrease in training performance but also a slight increase in generalisation performance. Eventually, this configuration proved to be the second best in terms of overall performance and the best in terms of generalisation performance.

6.3.1.7 *Configuration 7*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
3	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**2.24%**	6.42%	2.84%	10.75%	7.31%
Rentis House	23.11%	**19.95%**	12.65%	21.65%	21.90%
Pireaus House	24.64%	16.26%	**5.31%**	18.68%	18.36%
Kalamaki House	35.28%	17.74%	8.11%	**3.40%**	4.53%
Patras House	8.81%	1.76%	8.41%	28.57%	**1.57%**

Average Training Error	2.24%	13.18%	6.93%	13.62%	10.73%	9.34%
Average Generalisation Error	22.96%	11.92%	8.26%	28.57%		17.93%
Overall Error	18.82%	12.43%	7.47%	16.61%	10.73%	13.21%

Table 6.9 WL=3, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

In this experiment we modified the most successful configuration by reducing the number of weak learners added for each scene, similarly to Configuration 4. The rationale was to check whether less weak learners were adequate to capture user preferences. If this were the case, the advantage would be reduced computational complexity for the training and the simulation of the eventual ML Component configuration used in the actual system. The results revealed again a decrease in the training performance and a slight increase in the generalisation performance. In total, this has been the second best

6.3.1.8 Configuration 8

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
10	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**5.22%**	15.82%	6.12%	10.00%	6.12%
Rentis House	19.95%	**2.19%**	21.41%	10.95%	21.17%
Pireaus House	10.63%	13.85%	**9.02%**	11.76%	6.12%
Kalamaki House	11.32%	4.15%	8.30%	**2.08%**	3.77%
Patras House	9.39%	34.83%	9.39%	22.11%	**3.33%**

Average Training Error	5.22%	9.01%	12.18%	8.69%	8.10%	8.64%
Average Generalisation Error	12.82%	17.61%	8.85%	22.11%		15.35%
Overall Error	11.30%	14.17%	10.85%	11.38%	8.10%	11.16%

Table 6.10 WL=3, HL1=4, HL2=0, E=300, LR=0.02, M=0.3, EG=0.1

This configuration emphasised the weakness of the committee members and relied on their collective decision ability for successful classification. The results reveal a degradation of performance both in terms of training and generalisation error. A characteristic example is *Piraeus House* scene where the committee fails to improve its performance for the specific set of solutions despite the new members added to it. The reason is that new members, due to their low classification ability, feature less voting weight thus allowing the majority to misclassify the corresponding solutions.

6.3.1.9 Configuration 9

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.028	0.3	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**1.94%**	17.91%	2.69%	8.81%	6.87%
Rentis House	21.65%	**6.08%**	12.90%	14.84%	18.25%
Pireaus House	31.88%	12.56%	**4.35%**	13.04%	3.06%
Kalamaki House	43.21%	3.58%	18.68%	**1.32%**	9.43%
Patras House	10.76%	37.18%	10.57%	11.15%	**3.72%**

Average Training Error	1.94%	12.00%	6.64%	9.50%	8.27%	7.67%
Average Generalisation Error	26.88%	17.78%	14.62%	11.15%		17.61%
Overall Error	21.89%	15.46%	9.84%	9.83%	8.27%	13.06%

Table 6.11 WL=5, HL1=4, HL2=0, E=1000, LR=0.028, M=0.3, EG=0.1

Here again, we modify only one of the settings of the most efficient configuration up to this point by reducing the momentum parameter. The results are mediocre overall and training performance but, also, the second best generalisation performance.

6.3.1.10 Configuration 10

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.01	0.1	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**8.66%**	14.93%	4.33%	11.34%	8.06%
Rentis House	26.03%	**5.84%**	20.44%	17.27%	23.84%
Pireaus House	34.46%	24.32%	**6.12%**	12.40%	13.69%
Kalamaki House	43.77%	5.85%	9.81%	**3.02%**	7.55%
Patras House	9.00%	32.68%	2.35%	31.70%	**7.24%**

Average Training Error	8.66%	10.38%	10.30%	11.01%	12.08%	10.48%
Average Generalisation Error	28.32%	20.95%	6.08%	31.70%		21.76%
Overall Error	24.39%	16.72%	8.61%	15.15%	12.08%	15.39%

Table 6.12 WL=5, HL1=4, HL2=0, E=1000, LR=0.01, M=0.1, EG=0.1

Further weakening the committee members, by reducing the learning rate and the momentum, leads to reduced performance in every aspect. This configuration represents one of the lowest ranking combinations both in overall and generalisation error.

6.3.1.11 Configuration 11

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	300	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**6.12%**	17.76%	5.97%	10.90%	9.55%
Rentis House	22.38%	**8.52%**	21.17%	20.19%	18.25%
Pireaus House	19.81%	25.60%	**8.05%**	13.85%	6.60%
Kalamaki House	23.40%	9.62%	5.66%	**3.21%**	4.72%
Patras House	9.20%	41.49%	9.00%	30.72%	**4.31%**

Average Training Error	6.12%	13.14%	11.73%	12.04%	8.68%	10.34%
Average Generalisation Error	18.70%	25.57%	7.33%	30.72%		20.58%
Overall Error	16.18%	20.60%	9.97%	15.77%	8.68%	14.24%

Table 6.13 WL=5, HL1=4, HL2=0, E=300, LR=0.028, M=0.5, EG=0.1

Reducing only the epochs of the most efficient combination yields the current configuration. It is obvious that committee members are weakened and this contributes to the decrease of performance. This is an overall mediocre combination.

6.3.1.12 Configuration 12

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	300	0.028	0.5	0.3

	Error after training with solutions of scene				
Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**5.22%**	15.22%	5.67%	11.19%	18.21%
Rentis House	23.84%	**7.79%**	19.22%	18.73%	24.09%
Pireaus House	26.89%	20.93%	**5.64%**	20.93%	5.48%
Kalamaki House	25.66%	6.04%	10.57%	**4.15%**	14.15%
Patras House	8.22%	29.55%	9.20%	23.09%	**10.18%**

Average Training Error	5.22%	11.50%	10.18%	13.75%	14.42%	11.02%
Average Generalisation Error	21.15%	18.84%	9.88%	23.09%		18.24%
Overall Error	17.97%	15.91%	10.06%	15.62%	14.42%	14.79%

Table 6.14 WL=5, HL1=4, HL2=0, E=300, LR=0.028, M=0.5, EG=0.3

Despite the consensus regarding the 0.1 error goal [Lewitt03], [Polikar01], we have tried an average configuration with a higher error goal emphasising once again the weakness of the committee members. The configuration exhibits decreased performance when compared to the most efficient ones and produces similar results with the similar configuration featuring the 0.1 error goal.

6.3.1.13 Summary of Initial Settings Results

The following tables summarise the experimental results for the initial settings configuration. In particular, Table 6.15 presents the configuration ranking according to average training error, Table 6.16 presents configuration performance according to average generalisation error and Table 6.17 presents the overall ranking.

Exp. No.	*Weak Learners*	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*	*Average Training Error*	*Average Generalisation Error*	*Overall Error*
1	5	4	0	1000	0.028	0.5	0.1	6.53%	14.88%	10.93%
9	5	4	0	1000	0.028	0.3	0.1	7.67%	17.61%	13.06%
6	5	4	0	1000	0.09	0.5	0.1	8.44%	13.32%	11.90%
8	10	4	0	1000	0.028	0.5	0.1	8.64%	15.35%	11.16%
7	3	4	0	1000	0.028	0.5	0.1	9.34%	17.93%	13.21%
4	1	4	0	1000	0.028	0.5	0.1	9.60%	18.06%	12.05%
3	5	2	2	1000	0.028	0.5	0.1	10.23%	16.66%	14.92%
11	5	4	0	300	0.028	0.5	0.1	10.34%	20.58%	14.24%
10	5	4	0	1000	0.01	0.1	0.1	10.48%	21.76%	15.39%
2	5	4	0	300	0.15	0.5	0.1	10.71%	16.79%	14.16%
12	5	4	0	300	0.028	0.5	0.3	11.02%	18.24%	14.79%
5	5	4	0	300	0.028	0.9	0.1	13.96%	23.81%	19.13%

Table 6.15 Initial Settings in Descending Order of Training Performance

Exp. No.	*Weak Learners*	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*	*Average Training Error*	*Average Generalisation Error*	*Overall Error*
6	5	4	0	1000	0.09	0.5	0.1	8.44%	13.32%	11.90%
1	5	4	0	1000	0.028	0.5	0.1	6.53%	14.88%	10.93%
8	10	4	0	1000	0.028	0.5	0.1	8.64%	15.35%	11.16%
3	5	2	2	1000	0.028	0.5	0.1	10.23%	16.66%	14.92%
2	5	4	0	300	0.15	0.5	0.1	10.71%	16.79%	14.16%
9	5	4	0	1000	0.028	0.3	0.1	7.67%	17.61%	13.06%
7	3	4	0	1000	0.028	0.5	0.1	9.34%	17.93%	13.21%
4	1	4	0	1000	0.028	0.5	0.1	9.60%	18.06%	12.05%
12	5	4	0	300	0.028	0.5	0.3	11.02%	18.24%	14.79%
11	5	4	0	300	0.028	0.5	0.1	10.34%	20.58%	14.24%
10	5	4	0	1000	0.01	0.1	0.1	10.48%	21.76%	15.39%
5	5	4	0	300	0.028	0.9	0.1	13.96%	23.81%	19.13%

Table 6.16 Initial Settings in Descending Order of Generalisation Performance

Exp. No.	*Weak Learners*	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*	*Average Training Error*	*Average Generalisation Error*	*Overall Error*
1	**5**	**4**	**0**	**1000**	**0.028**	**0.5**	**0.1**	**6.53%**	**14.88%**	**10.93%**
8	**10**	**4**	**0**	**1000**	**0.028**	**0.5**	**0.1**	**8.64%**	**15.35%**	**11.16%**
6	**5**	**4**	**0**	**1000**	**0.09**	**0.5**	**0.1**	**8.44%**	**13.32%**	**11.90%**
4	**1**	**4**	**0**	**1000**	**0.028**	**0.5**	**0.1**	**9.60%**	**18.06%**	**12.05%**
9	5	4	0	1000	0.028	0.3	0.1	7.67%	17.61%	13.06%
7	3	4	0	1000	0.028	0.5	0.1	9.34%	17.93%	13.21%
2	5	4	0	300	0.15	0.5	0.1	10.71%	16.79%	14.16%
11	5	4	0	300	0.028	0.5	0.1	10.34%	20.58%	14.24%
12	5	4	0	300	0.028	0.5	0.3	11.02%	18.24%	14.79%
3	5	2	2	1000	0.028	0.5	0.1	10.23%	16.66%	14.92%
10	5	4	0	1000	0.01	0.1	0.1	10.48%	21.76%	15.39%
5	5	4	0	300	0.028	0.9	0.1	13.96%	23.81%	19.13%

Table 6.17 Initial Settings in Descending Order of Overall Performance

Next, we present a chart graph of the evolution of overall error for all configurations where it becomes apparent that, apart from the best performing configurations, there are others that have exhibited a consistently reducing overall error as, for example, the ones used in Configuration 5 and Configuration 9.

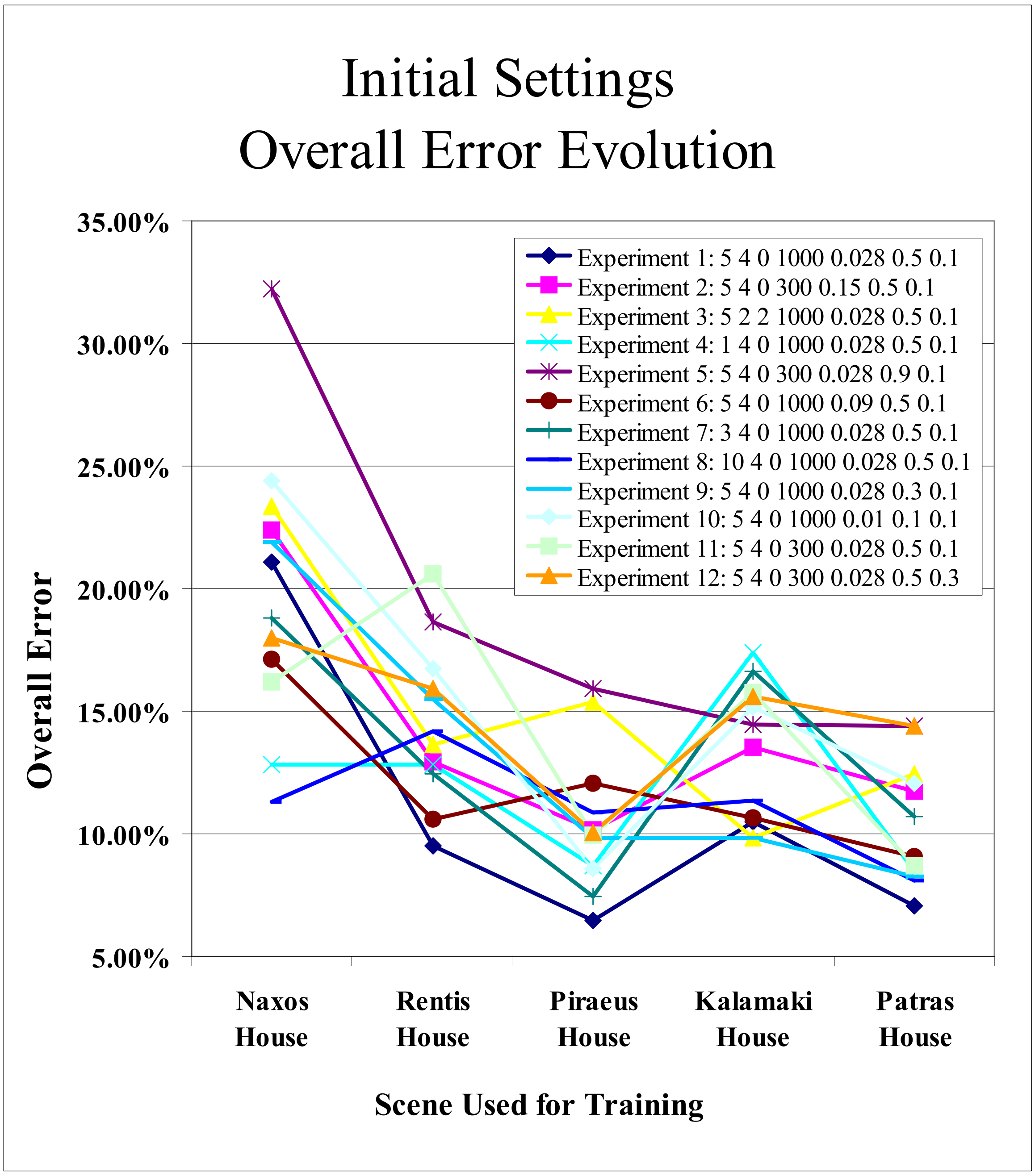


Figure 6.11 Evolution of Overall Error for Each Configuration

6.4 Module Evaluation

After the initial settings evaluation was concluded, a series of experiments took place, this time based on actual user's profiles and realistic solutions evaluations. The solutions for the aforementioned example scenes had already been generated and mapped to the observed attributes; a series of alternative Decision Support user profiles was created, reflecting different users' preferences, in order to be used for automatic solution evaluation. Notice that the same set of solutions, already mapped to the observed attributes, may be evaluated differently according to different users' Decision Support profiles. The different Decision Support profiles include alternative middle values that modify not only the overall solution grades, due to different weight assignments, but also different grades per attribute due to these alternative middle values as was explained in the previous chapter. The alternative Decision Support profiles appear in Table 6.18.

		Observed Attributes						
		public zone area	***private zone area***	***number of sleep rooms***	***number of wash rooms***	***south-western bedroom***	***public-private zone separation***	***percentage of non-oblong rooms***
Pangiotis	***Middle values***	6	9	1	1	0	0	80
	AHP weights	0.427	0.306	0.041	0.045	0.032	0.109	0.041
	SMART weights	0.345	0.276	0.035	0.035	0.069	0.172	0.069
	RR weights	0.386	0.193	0.064	0.077	0.055	0.129	0.096
Marianna	***Middle values***	7	10	2	1	0	0	75
	AHP weights	0.304	0.253	0.155	0.054	0.030	0.147	0.056
	SMART weights	0.294	0.265	0.177	0.088	0.029	0.088	0.059
	RR weights	0.386	0.193	0.129	0.064	0.055	0.096	0.077
Giorgos	***Middle values***	4	14	3	2	0	0	50
	AHP weights	0.122	0.116	0.025	0.025	0.230	0.232	0.251
	SMART weights	0.125	0.125	0.025	0.025	0.225	0.250	0.225
	RR weights	0.096	0.077	0.064	0.055	0.129	0.386	0.193

Table 6.18 Actual users' Decision Support profiles

6.4.1.1 *Configuration 1a: User=Panagiotis*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House	
Naxos House	**16.57%**	11.64%	17.91%	17.16%	14.48%	
Rentis House	6.57%	**5.60%**	5.35%	9.73%	16.30%	
Pireaus House	1.45%	0.16%	**0.48%**	0.97%	6.28%	
Kalamaki House	2.64%	2.83%	2.64%	**2.83%**	3.58%	
Patras House	30.33%	19.18%	17.22%	16.63%	**23.48%**	
Average Training Error	16.57%	8.62%	7.92%	7.67%	12.83%	10.72%
Average Generalisation Error	10.25%	7.39%	9.93%	16.63%		11.05%
Overall Error	11.51%	7.88%	8.72%	9.47%	12.83%	10.08%

Table 6.19 User=Panagiotis, WL=5, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

In this first experiment of core module evaluation we have used the most efficient configuration of the Machine Learning Component in an attempt to capture and maintain a particular user's preferences. We have expected worse results than the initial settings evaluations mostly due to the virtually unpredictable and not necessarily consistent nature of user preferences. This, however, has not always been the case. The module generalises acceptably for certain descriptions, even after the first training session, and poorly for some others even after a number of scenes. The overall results are promising regarding the dependence to the particular user's preferences.

6.4.1.2 *Configuration 1b: User=Marianna*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**14.18%**	17.46%	18.21%	16.57%	28.66%
Rentis House	9.73%	**15.33%**	31.14%	23.36%	21.17%
Pireaus House	5.96%	2.09%	**0.16%**	0.00%	6.28%
Kalamaki House	1.89%	2.45%	6.04%	**1.70%**	3.58%
Patras House	26.61%	31.51%	31.31%	31.31%	**35.42%**

Average Training Error	14.18%	16.40%	16.50%	10.41%	19.02%	15.30%
Average Generalisation Error	11.05%	12.02%	18.67%	31.31%		18.26%
Overall Error	11.67%	13.77%	17.37%	14.59%	19.02%	15.28%

Table 6.20 User=Marianna, WL=5, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

The same configurations do not yield necessarily the same performance in the current context due to the difference in the nature of user's preferences and their relation to each scene's intricacies. In the specific example we have the case where the Machine Learning Component yields a better overall error after training with only the first scene than in all subsequent cases. This is mostly due to the fact that the user evaluation seems to represent contradictive classifications that mislead the committee majority. The effort of the module to take into account new training subsets degrades overall training performance.

6.4.1.3 Configuration 1c: User=Giorgos

Weak Learners	Hidden Layer 1 Neurons	Hidden Layer 2 Neurons	Epochs	Learning Rate	Momentum	Error Goal
5	4	0	1000	0.028	0.5	0.1

	Error after training with solutions of scene					
Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House	
Naxos House	**6.72%**	5.37%	6.27%	5.52%	6.12%	
Rentis House	12.90%	**8.52%**	8.52%	8.27%	10.22%	
Pireaus House	10.14%	8.53%	**6.28%**	7.25%	7.57%	
Kalamaki House	3.40%	3.21%	3.21%	**3.21%**	3.40%	
Patras House	14.09%	9.39%	8.41%	8.02%	**9.39%**	
Average Training Error	6.72%	6.94%	7.02%	6.06%	7.34%	6.82%
Average Generalisation Error	10.13%	7.05%	5.81%	8.02%		7.75%
Overall Error	9.45%	7.00%	6.54%	6.45%	7.34%	7.36%

Table 6.21 User=Giorgos, WL=5, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

In this experiment user preferences are efficiently captured by the ML Component as demonstrated by the low error rates in the overall. Nevertheless, for the last scene, the training has the reverse result, increasing the training error. Similar to the previous experiment, user evaluation for the new training sets leads the committee members to contradictive evaluations in which case, the voting weights suggest the, not always correct, overall evaluation. Notice, however, the varied overall error response of the ML Component using the same configuration for different users' solution evaluations.

6.4.1.4 *Configuration 4a: User=Panagiotis*

Weak Learners	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*
1	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House	
Naxos House	**8.21%**	8.21%	16.27%	19.25%	15.82%	
Rentis House	10.95%	**10.95%**	10.71%	14.84%	11.92%	
Pireaus House	1.77%	1.77%	**0.32%**	2.09%	1.77%	
Kalamaki House	2.64%	2.64%	4.53%	**4.15%**	2.26%	
Patras House	19.96%	19.96%	26.22%	18.20%	**14.87%**	
Average Training Error	8.21%	9.58%	9.10%	10.08%	9.33%	9.26%
Average Generalisation Error	8.83%	8.12%	15.38%	18.20%		12.63%
Overall Error	8.71%	8.71%	11.61%	11.71%	9.33%	10.01%

Table 6.22 User=Panagiotis, WL=1, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

The error evolution in this experiment resembles that of the similar experiments of the previous section covering the initial settings evaluation. In particular, each sub-committee contains only a single member which, due to the low error and hence high voting weight of the first member, fails to impose its evaluation in the overall outcome. Thus, once again only after the third single-member sub-committee is added to the overall committee the results start to vary. The same is true for the next two users' experiments based on the same configuration.

6.4.1.5 Configuration 4b: User=Marianna

Weak Learners	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*
1	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**4.18%**	4.18%	11.94%	4.18%	12.84%
Rentis House	8.52%	**8.52%**	9.25%	8.52%	12.90%
Pireaus House	2.42%	2.42%	**0.16%**	2.42%	2.74%
Kalamaki House	2.26%	2.26%	2.26%	**2.08%**	2.45%
Patras House	15.85%	15.85%	31.31%	15.66%	**17.61%**

Average Training Error	4.18%	6.35%	7.12%	4.30%	9.71%	6.33%
Average Generalisation Error	7.26%	6.84%	16.79%	15.66%		11.64%
Overall Error	6.65%	6.65%	10.98%	6.57%	9.71%	8.11%

Table 6.23 User=Marianna, WL=1, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

6.4.1.6 *Configuration 4c: User=Giorgos*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
1	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**5.97%**	5.97%	6.12%	6.12%	6.27%
Rentis House	10.46%	**10.46%**	8.76%	8.76%	10.71%
Pireaus House	10.47%	10.47%	**7.57%**	7.57%	7.57%
Kalamaki House	3.77%	3.77%	3.21%	**3.21%**	3.40%
Patras House	13.31%	13.31%	10.18%	10.18%	**10.76%**

Average Training Error	5.97%	8.22%	7.48%	6.41%	7.74%	7.16%
Average Generalisation Error	9.50%	9.18%	6.69%	10.18%		8.89%
Overall Error	8.80%	8.80%	7.17%	7.17%	7.74%	7.93%

Table 6.24 User=Giorgos, WL=1, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

6.4.1.7 *Configuration 6a: User=Panagiotis*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.09	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**14.03%**	14.18%	17.61%	14.63%	14.78%
Rentis House	17.76%	**6.08%**	8.03%	12.41%	9.73%
Pireaus House	6.28%	0.16%	**0.00%**	2.74%	2.42%
Kalamaki House	4.34%	4.15%	2.26%	**1.13%**	1.32%
Patras House	26.22%	21.33%	26.42%	22.11%	**13.70%**

Average Training Error	14.03%	10.13%	8.55%	7.73%	8.39%	9.76%
Average Generalisation Error	13.65%	8.55%	14.34%	22.11%		14.66%
Overall Error	13.73%	9.18%	10.86%	10.60%	8.39%	10.55%

Table 6.25 User=Panagiotis, WL=5, HL1=4, HL2=0, E=1000, LR=0.09, M=0.5, EG=0.1

Although this configuration does not yield the best training or generalisation error, it exhibits a promising performance, by reducing the training error of the current training scene while generally improving the training error and the average generalisation. This smooth error reduction is also shown in the chart appearing in Figure 6.12.

6.4.1.8 *Configuration 6b: User=Marianna*

Weak Learners	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*
5	4	0	1000	0.09	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House	
Naxos House	**10.15%**	10.00%	12.69%	15.97%	14.93%	
Rentis House	8.03%	**6.57%**	27.01%	11.44%	13.38%	
Pireaus House	2.42%	1.61%	**0.00%**	2.25%	5.80%	
Kalamaki House	1.51%	3.21%	5.66%	**1.51%**	3.21%	
Patras House	19.96%	15.46%	9.98%	20.35%	**3.13%**	
Average Training Error	10.15%	8.28%	13.23%	7.79%	8.09%	9.51%
Average Generalisation Error	7.98%	6.76%	7.82%	20.35%		10.73%
Overall Error	8.41%	7.37%	11.07%	10.30%	8.09%	9.05%

Table 6.26 User=Marianna, WL=5, HL1=4, HL2=0, E=1000, LR=0.09, M=0.5, EG=0.1

Once again, the effort to integrate the knowledge suggested by the evaluated solutions corresponding to a new scene lead to performance degradation as is the case with the Pireaus House solutions. In particular, the ML Component perfectly captures user preferences for the specific solutions at the cost of losing a considerable amount of previous knowledge, especially for the Rentis House scene. However, as the experiment continues, the balance is restored. Once again we have to notice that actual user preferences may not follow a consistent pattern thus making generalisation a tough task, as it becomes apparent in the case of the Patras House in the specific experiment.

6.4.1.9 Configuration 6c: User=Giorgos

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
5	4	0	1000	0.09	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**5.67%**	6.27%	6.57%	5.52%	6.12%
Rentis House	11.19%	**8.03%**	11.92%	11.44%	8.03%
Pireaus House	16.26%	7.57%	**16.26%**	10.79%	7.41%
Kalamaki House	3.58%	3.40%	3.21%	**3.21%**	3.21%
Patras House	16.83%	10.96%	19.18%	16.44%	**8.41%**

Average Training Error	5.67%	7.15%	11.58%	7.74%	6.64%	7.76%
Average Generalisation Error	11.97%	7.31%	11.19%	16.44%		11.73%
Overall Error	10.71%	7.24%	11.43%	9.48%	6.64%	9.10%

Table 6.27 User=Giorgos, WL=5, HL1=4, HL2=0, E=1000, LR=0.09, M=0.5, EG=0.1

It is worth noticing that Piraeus House, the same scene that is perfectly captured in the last two experiments, causes considerable error increase in the current case revealing once again the dynamic nature of the actual user's preferences and its effect on the module's response.

6.4.1.10 Configuration 8a: User=Panagiotis

Weak Learners	*Hidden Layer 1 Neurons*	*Hidden Layer 2 Neurons*	*Epochs*	*Learning Rate*	*Momentum*	*Error Goal*
10	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**10.15%**	12.24%	19.85%	14.03%	15.07%
Rentis House	17.52%	**5.60%**	8.27%	9.73%	17.76%
Pireaus House	5.64%	0.16%	**0.32%**	0.16%	6.28%
Kalamaki House	3.58%	1.32%	3.21%	**0.75%**	4.34%
Patras House	36.20%	21.72%	37.77%	24.46%	**12.13%**

Average Training Error	10.15%	8.92%	9.48%	6.17%	11.12%	9.17%
Average Generalisation Error	15.74%	7.73%	20.49%	24.46%		17.11%
Overall Error	14.62%	8.21%	13.88%	9.83%	11.12%	11.53%

Table 6.28 User=Panagiotis, WL=10, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

The increased number of weak learners per scene leads to a committee of strong majorities that fails to generalise effectively with respect to the Patras House scene. However, the incorporation of new solutions generally improves training error except in the case of the last, obviously harder to learn, set of user evaluated solutions corresponding to the Patras House description.

6.4.1.11 Configuration 8b: User=Marianna

Weak Learners	Hidden Layer 1 Neurons	Hidden Layer 2 Neurons	Epochs	Learning Rate	Momentum	Error Goal
10	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**6.72%**	12.39%	12.69%	16.27%	12.39%
Rentis House	8.27%	**6.81%**	4.14%	7.30%	12.65%
Pireaus House	3.06%	0.16%	**0.00%**	1.93%	6.44%
Kalamaki House	2.64%	1.13%	0.94%	**1.32%**	2.83%
Patras House	13.70%	18.59%	18.59%	19.77%	**2.54%**

Average Training Error	6.72%	9.60%	5.61%	6.71%	7.37%	7.20%
Average Generalisation Error	6.92%	6.63%	9.77%	19.77%		10.77%
Overall Error	6.88%	7.82%	7.27%	9.32%	7.37%	7.73%

Table 6.29 User=Marianna, WL=10, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

The ML Component exhibits impressive generalisation ability for most of the scenes and effectively learns each submitted scene. Notice how the last scene is incorporated in the committee's knowledge at a relatively low expense with respect to the error regarding previously seen scenes.

6.4.1.12 *Configuration 8c: User=Giorgos*

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***
10	4	0	1000	0.028	0.5	0.1

Error after training with solutions of scene

Scene ID	Naxos House	Rentis House	Pireaus House	Kalamaki House	Patras House
Naxos House	**9.85%**	5.97%	12.54%	6.12%	7.16%
Rentis House	39.66%	**8.03%**	17.03%	7.79%	9.98%
Pireaus House	32.05%	7.57%	**25.44%**	7.57%	7.73%
Kalamaki House	6.79%	3.21%	5.85%	**3.21%**	3.40%
Patras House	20.35%	9.78%	20.55%	10.96%	**11.35%**

Average Training Error	9.85%	7.00%	18.34%	6.17%	7.92%	9.86%
Average Generalisation Error	24.71%	6.85%	13.20%	10.96%		13.93%
Overall Error	21.74%	6.91%	16.28%	7.13%	7.92%	12.00%

Table 6.30 User=Giorgos, WL=10, HL1=4, HL2=0, E=1000, LR=0.028, M=0.5, EG=0.1

Once again different users' preferences lead to different behaviour. In the particular experiment the overall generalisation performance is satisfactory, yet each scene is accepted with varying responses, ranging from successful incorporation to the committee's knowledge (Rentis House), indifference (Patras House) or performance degradation for the specific scene (Piraeus House).

6.4.1.13 Comparison with Decision Support Profiles

In the following we summarise the compared performances of the Machine Learning Profiles for the aforementioned users based on the configurations presented above and the corresponding Decision Support Profiles. In order to maintain compatibility, presents the evolution of error after each scene was used for training of the Machine Learning Component although the corresponding evaluation on behalf of the Decision Support Component was already available as soon as the solutions had been generated, without any need for training. The chart in Figure 6.12 explicitly shows the performance dominance of the Machine Learning Profiles in general at least for the specific user.

User: Panagiotis		***Error after training with scene***				
		Naxos House	***Rentis House***	***Pireaus House***	***Kalamaki House***	***Patras House***
Decision Support Profile Error	***AHP***	21.94%	20.22%	16.16%	13.25%	18.08%
	SMART	23.73%	21.05%	18.92%	16.05%	19.55%
	RR	23.13%	21.39%	19.83%	17.02%	20.00%
Machine Learning Profile Error	***Configuration1***	16.57%	8.62%	7.92%	7.67%	12.83%
	Configuration 4	8.21%	9.58%	9.10%	10.08%	9.33%
	Configuration 6	14.03%	10.13%	8.55%	7.73%	8.39%
	Configuration 8	10.15%	8.92%	9.48%	6.17%	11.12%

Table 6.31 Comparison of error evolution for alternative Decision Support Machine Learning Profiles for User Panagiotis

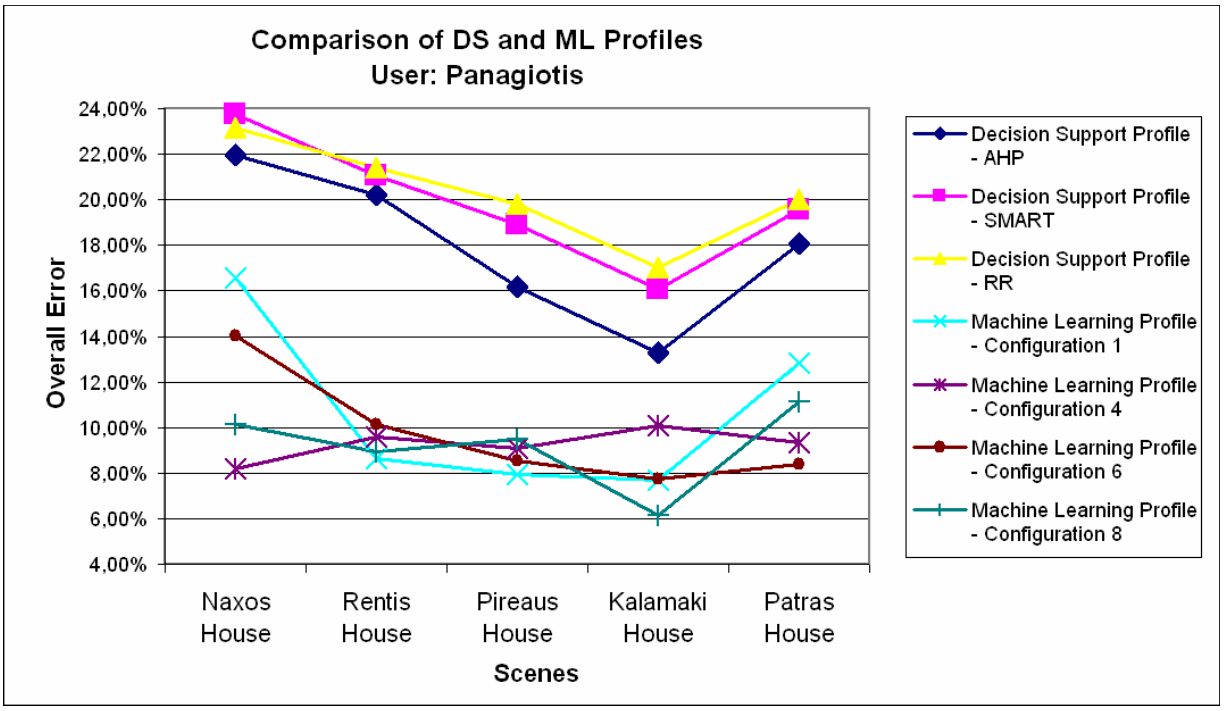


Figure 6.12 Comparison of DS and ML Profiles for User Panagiotis

User: Marianna		***Error after training with scene***				
		Naxos House	***Rentis House***	***Pireaus House***	***Kalamaki House***	***Patras House***
Decision Support Profile Error	***AHP***	21.94%	20.22%	16.16%	13.25%	18.08%
	SMART	23.73%	21.05%	18.92%	16.05%	19.55%
	RR	23.13%	21.39%	19.83%	17.02%	20.00%
Machine Learning Profile Error	***Configuration1***	14.18%	16.40%	16.50%	10.41%	19.02%
	Configuration 4	4.18%	6.35%	7.12%	4.30%	9.71%
	Configuration 6	10.15%	8.28%	13.23%	7.79%	8.09%
	Configuration 8	6.72%	9.60%	5.61%	6.71%	7.37%

Table 6.32 Comparison of error evolution for alternative Decision Support Machine Learning Profiles for User Marianna

It is interesting to observe that, in the case of the following user, the DS Profiles generally outperform one of the most efficient initial settings configurations for the ML

Component, thus proving the efficiency of the DS Profiles alternative, especially when considering that the latter only requires initialisation on behalf of the user and no further training. Of course, other configurations of the ML Profile still prove more efficient even for the specific user.

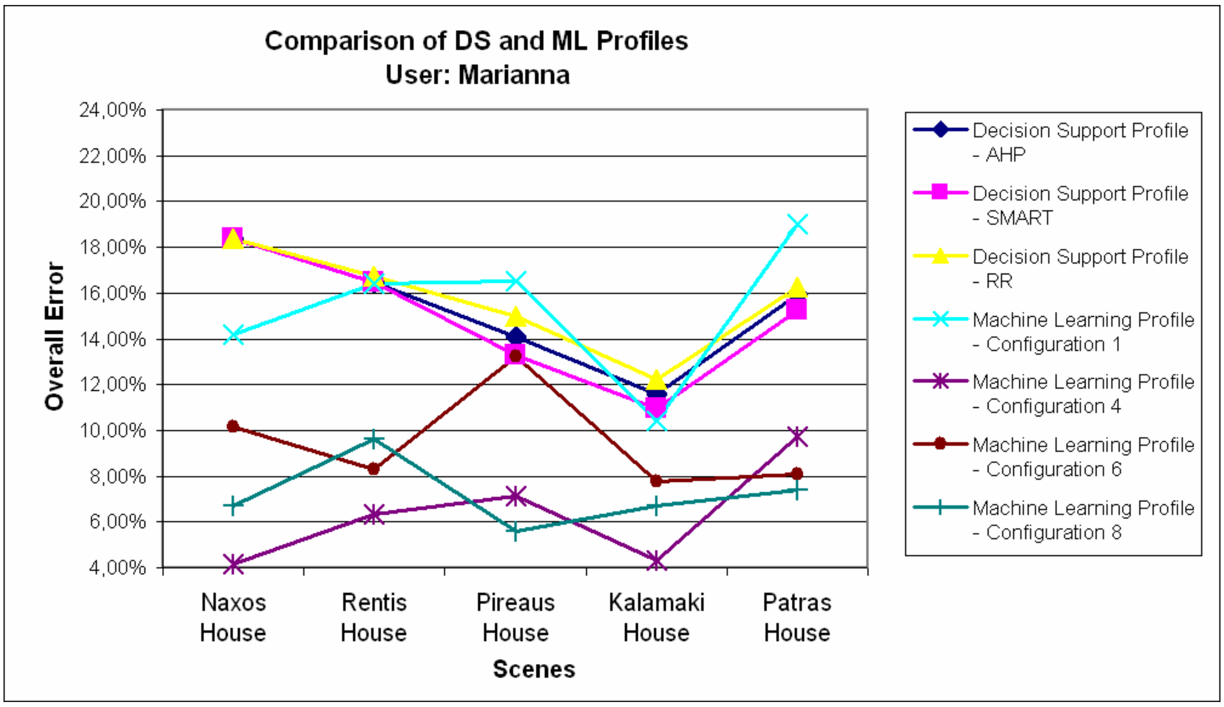


Figure 6.13 Comparison of DS and ML Profiles for User Marianna

User: Giorgos		***Error after training with scene***				
		Naxos House	***Rentis House***	***Pireaus House***	***Kalamaki House***	***Patras House***
Decision Support Profile Error	***AHP***	17.61%	24.50%	25.67%	20.06%	20.19%
	SMART	17.76%	24.57%	25.72%	20.09%	20.18%
	RR	25.67%	34.98%	34.21%	26.79%	25.62%
Machine Learning Profile Error	***Configuration1***	1.94%	7.58%	11.87%	2.90%	6.33%
	Configuration 4	4.48%	11.24%	7.23%	9.25%	4.71%
	Configuration 6	5.37%	2.69%	2.62%	6.50%	19.91%
	Configuration 8	11.04%	10.53%	4.63%	3.44%	2.08%

Table 6.33 Comparison of error evolution for alternative Decision Support Machine Learning Profiles for User Giorgos

In this case we may observe that the SMART and AHP DS Profiles exhibit practically identical performance as opposed to the RR based DS Profile that here fails to adequately capture user's preferences. The ML Profiles once again outperform the DS counterparts thus suggesting an overall dominance of the former.

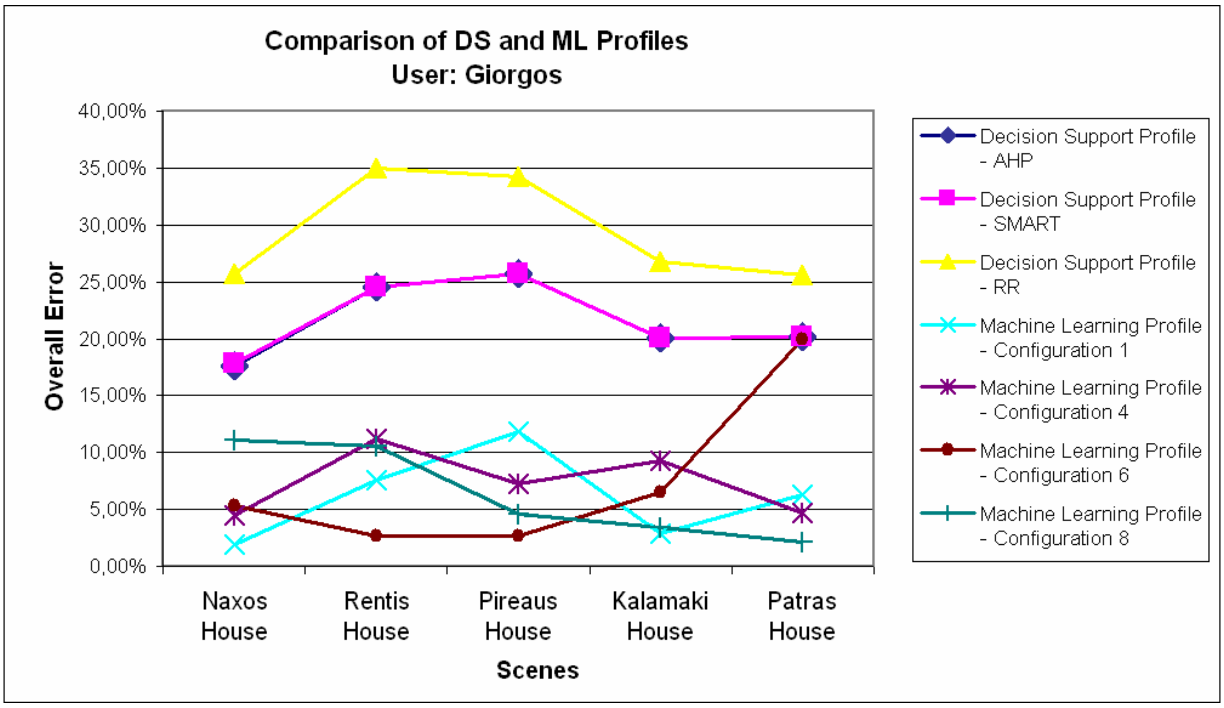


Figure 6.14 Comparison of DS and ML Profiles for User Giorgos

6.5 Fine Tuning

All experiments up to now have suggested dominance of the Machine Learning Profiles in most cases. Nevertheless, in an effort to cover all special cases, mostly with respect to user's preferences that may not always be consistent, we have performed an additional series of experiments for the special case where the approved solutions are much less than the total solutions inspected by the user. In such a case, we have to employ the performance indices presented in the end of the previous chapter in order to capture the intricate details of the behaviour of each configuration of the Machine Learning Component. Intuitively, when only a few approved solutions exist, it is harder for a mechanism to classify them correctly and avoid discarding the whole training set. The *Hit Rate,* formally defined in the previous chapter, reveals how successful a configuration is with respect to its own approved solutions. In other words, a high Hit Rate implies that most of the solutions approved by the user are also approved by the configuration. Using the actual solution evaluations from user Giorgos we obtain the following statistics regarding the approved solutions, appearing in Table 6.34, suggesting that can be used as the basis for this stage of the experiments.

	Naxos House	***Rentis House***	***Piraeus House***	***Kalamaki House***	***Patras House***
Approved	41	33	47	17	52
Inspected	670	411	621	530	511
Percentage	6.12%	8.03%	7.57%	3.21%	10.18%

Table 6.34 Approved/Inspected Solutions Percentage

Only small percentages of the solutions have been approved by the actual user. Hence we have performed an additional series of experiments for various configurations aiming explicitly to maximise Hit Rate. These experiments appear in Table 6.35 (only Naxos House training is presented) together with the performance of one of the already examined configurations from the previous section appearing in the first row.

Weak Learners	***Hidden Layer 1 Neurons***	***Hidden Layer 2 Neurons***	***Epochs***	***Learning Rate***	***Momentum***	***Error Goal***	***Hit Rate***
10	**4**	**0**	**1000**	**0.028**	**0.5**	**0.1**	**65.85%**
10	4	0	1000	0.09	0.5	0.1	36.59%
10	4	0	1000	0.15	0.5	0.1	41.46%
10	**4**	**0**	**1000**	**0.2**	**0.5**	**0.1**	**92.68%**
5	4	0	1000	0.18	0.5	0.1	82.93%
10	4	0	1000	0.18	0.5	0.1	90.24%
10	4	0	300	0.2	0.5	0.1	80.49%
5	4	0	1000	0.2	0.5	0.1	80.49%
10	4	0	1000	0.25	0.5	0.1	90.24%

Table 6.35 Machine Learning Component configurations maximising Hit Rate after training with the Naxos House scene

From these experiments we have selected the configuration exhibiting the largest Hit Rate for a comparison with one of the most efficient configurations of the entire experiment series as well as with the Decision Support Profiles of the specific user. At this stage, the comparison is based on the performance indices defined in the previous chapters.

User: Giorgos		***Error***	***Hit Rate***	***Performance Ratio***	***Miss Rate***	***Modified Miss Rate***
Decision Support Profile Indices	***AHP***	20.19%	58.29%	16.94%	19.45%	2.29
	SMART	20.18%	58.29%	16.95%	19.44%	2.29
	RR	25.62%	69.30%	15.94%	26.12%	3.19
Machine Learning Profile Indices	***New Configuration after all scenes***	6.26%	10.51%	70.98%	0.19%	0.01
	Configuration 6 after all scenes	7.92%	25.64%	30.21%	3.49%	0.37

Table 6.36 Comparison of DS and ML configurations based on performance indices

Both Machine Learning Component configurations exhibit extremely low Error levels at the end of this experiment. Moreover, *Configuration 6* maintains satisfactory levels for all indices regarding the fact that only a small percentage of the solutions were actually approved by the user. The Decision Support Profiles demonstrate high Hit Rates, despite the high Error levels. This is mainly due to their construction that leads to the approval of a larger number of solutions than the Machine Learning Component similarly to one of the extreme cases in the discussion of the performance indices in the previous chapter. This also becomes apparent from the increased Miss Rates and Modified Miss Rates, suggesting increased numbers of erroneously approved solutions. Notice how the promising high Hit Rate of the new configuration has been severely decreased after training with more scenes. Nevertheless, the *New Configuration* exhibits increased Performance Ratio but a low Hit Rate, implying that it approves only a few solutions but most of them are also approved by the user. Hence, it seems more fitting to a *solution search* mode of the declarative design environment.

Conclusion

7 Conclusions

In this work we have combined and adapted concepts originating from the Multicriteria Decision Analysis and Machine Learning areas and have applied them to the context of Declarative Modelling methodologies. Our aim has been to enrich an already existing environment of Declarative Design with intelligent features that improve the response of this environment to any user's custom preferences. The exchange of ideas among the aforementioned areas and the adaptation and application of specific concepts to the prototype that has been built have provided new insight and fruitful results in the area of declarative modelling. On the other hand, our work has yielded interesting results regarding the flexibility of the decision analysis and machine learning techniques used and the degree of adaptability they offer when transferred to another domain.

7.1 Comparison with Other Works

Despite the wide range of works concerned with user preference acquisition and subsequent application in various fields, only a few have been recorded in the area of declarative modelling. In particular, [Plemenos02] represents such an effort in the current context whereas [Champciaux98a] addresses the same problem in a very similar environment. Nevertheless, as it is explained in detail in this section, both of these works operate under a different set of assumptions thus yielding a different form of user preference modelling.

In the [Plemenos02] proposal, two alternative approaches have been implemented towards user preference modelling. According to the first, the structure of each submitted declarative scene is mapped to a dedicated neural network. This network is subsequently trained with the user approved scenes in order to acquire the user's interpretation of the declarative relations and properties comprising the specific description. After the training period, the network is able to evaluate and present to the user only the scenes that comply with his/her preferences. Similarly to our approach, the proposed system does not reduce the number of solutions generated by the system since it operates on the geometric representations produced by the solution generator. In fact, the authors mention as a future extension the possibility of applying the neural network's evaluation *during* solution generation, thus preventing a number of solutions from being explored by the constraint satisfaction mechanism performing solution generation. Up to this date, this idea has not been further explored.

The main difference of the Intelligent User Profile Module proposed and implemented in the current work when compared with the aforementioned proposal originates from the *description-oriented* nature of the machine learning mechanism used. In particular, the aforementioned methodology accurately captures and maintains user preferences for each declarative description separately. Practically, after a number of sessions, the user will have submitted a number of diverse declarative descriptions, each represented by its own neural network, able to recognise *user approved* solutions for the corresponding description. However, this configuration implies that, as soon as a new description is submitted by the user, the previously acquired knowledge regarding his/her preferences can not be exploited with respect to the newly generated solutions. In other words, the proposed configuration is able to *generalise* its knowledge only within the limits of *previously seen* declarative descriptions. On the other hand, due to the construction of the Intelligent User Profile Module proposed in the current work, previously acquired knowledge is fully exploitable in order to evaluate solutions originating from a new declarative description. Of course, even in our case, and as it became apparent during the experiments, *generalisation is not always successful* since each description may reveal different aspects of the user's preferences. However, in a considerable number of experiments, our mechanism was able to adequately generalise, successfully selecting user approved solutions among a newly generated set. This is also due to the core assumption we have made that each user is interested in a specific set of observed attributes, which, in turn, offer a certain degree of *normalisation* among solutions generated by different descriptions. In a sense, the mechanism proposed in [Plemenos02] addresses the user preference problem *vertically,* within the limits of one declarative description at a time, while we address the same problem *horizontally,* across the entire range of past as well as future declarative descriptions potentially submitted by the user.

The alternative methodology proposed in [Plemenos02] is based on genetic techniques. In particular, an initial solution population is generated and gradually improved through the standard genetic operations of crossover, mutation and cloning. Evaluation of solutions by the user after inspection plays the role of the fitness function required in the genetic techniques context. The main difference of this approach is that it serves mainly as a *solution search* generation mechanism relying on repetitive user feedback. Unlike the case of the previous mechanism, not all solutions complying with the input declarative description are generated. This implies faster system response as an advantage at the expense of the requirement for considerable user intervention and the possibility that certain (possibly highly preferable) solutions may never be generated – due to the localised search nature of the genetic techniques. On the other hand, the Intelligent User Profile Module proposed and implemented in the current work requires minimal user intervention and guarantees discovery of the highest preferable solutions at the expense of slower overall system response. Nevertheless, the comparison between the two approaches is not entirely valid since one represents a *solution search* technique whereas the other operates in *exploration* mode, both discussed in the Declarative Modelling chapter. In fact, the aforementioned advantages and disadvantages represent the result of the alternative interpretations of the trade-off implied by the two declarative modelling modes.

The approach proposed in [Champciaux98a] comprises two main modules: an unsupervised classifier, assigning solutions corresponding to each declarative description to an automatically generated class hierarchy based on similarity, using the Cobweb algorithm [Fisher87] for this purpose, and a user interface module that allows the user to inspect and evaluate class representatives thus training a supervised machine learning mechanism to recognise a target concept, i.e. a more restrictive interpretation of the initial description. The mechanism relies on the assumption that similar solutions share a similar degree of preference on behalf of the user. Moreover, the latter of the aforementioned modules implies operation in a mode closer to *solution search* in the sense that entire classes of similar solutions are eliminated through user evaluation to yield a small subset of geometrically similar solutions representing the target concept. Last but not least, the concept hierarchy and the target concept defined by the user are defined based on a single declarative description at a time thus revealing a *vertical* consideration of the user preferences. Considering the notion of the target concept similar to that of user preferences, the Intelligent User Profile Module proposed and implemented in the current work provides far greater flexibility in their definition, in the sense that the user is free to suggest different evaluations for otherwise similar solutions through the Machine Learning Component whereas the Decision Support Component can still be used for similar solution classification. Once again, the most important difference is that a common model for user preferences, that is acquired and maintained through the evaluation of alternative declarative descriptions, can be instantly applied to any newly generated solutions originating from a previously unseen declarative description. In other words, the two approaches serve a different purpose, the one in [Champciaux98a] aiming to refine the submitted declarative description through classification and user feedback, whereas the one presented in this work is aiming to capture an overall model of a particular user's preferences through regular system use.

7.2 Declarative Modelling Benefits

The motivation for this work stems from the problem of combinatorial explosion that is inherent to Declarative Modelling. Modest descriptions containing a few objects and relations among them may yield hundreds of thousands of alternative valid solutions. The capabilities of modern day equipment have made it possible to generate all alternative solutions within a fraction of the time once needed for the same task. However, the capability of humans to review these solutions and apply their judgement in order to select those closer to their preferences has remained more or less the same through the years. This is one of the reasons we have set the task of intelligent solution evaluation as the main focus of our research.

7.2.1 Improved Solution Population

Declarative Modelling offers the user the ability to start with an abstract, high level description of an object, in our context a building scene, relieving him/her from the need to specify its concrete geometric properties. Moreover, the fact that the input description is abstract and, up to a certain degree, ambiguous may lead to alternative solutions that may be of interest to the user, although not originally conceived by him/her as part of the results.

The incorporation of the Intelligent User Profile Module to the Open-MultiCAD environment offers the user the advantage of receiving a subset of the solution population which is closer to his/her personal preferences. Inspection of all generated solutions is still possible at the user's discretion since the module presents the generated solutions in descending preference order. In this manner, the user who is not willing to inspect all generated solutions may choose to stop solution viewing after a few representatives, knowing that the solutions he/she has seen are those closest to his/her preferences. This fact actually represents an alternative approach to *solution search* mode, discussed in the Declarative Modelling chapter earlier, for the Open-MultiCAD environment. The user is indeed presented, in a transparent manner, with the best representative(s) of the solution space corresponding to the declarative description of the scene submitted as input. This corresponds exactly to the notion of solution search mode and, in the current context, solution quality when considering the *best,* is connected with the specific user's preferences.

7.2.2 Minimal Overhead

The essence of the declarative modelling is the ability of the user to describe a scene in an abstract form, thus avoiding concrete geometric details. In compliance with this we have respected, throughout this work, this abstract form of input since it forms a fundamental element of declarative modelling. Hence, we have avoided requiring any additional input regarding each scene and we have selected to focus the operation of the transparent mechanism on the geometric representation of the solutions.

The only additional input required by the user comprises a *pre-processing* task, that takes place *once for every user*, where he/she answers a set of questions regarding his/her preferences in order to initialise the corresponding *decision support profile*. The questions refer to the importance of the observed characteristics and do not require explicit geometric information.

After this initialisation the user proceeds with the regular use of the system, submitting declarative descriptions and evaluating solutions as he/she would have done normally. Even at that stage, our main concern has been to avoid the introduction any additional overhead for the user with respect to solution evaluation. The reason is that such an overhead would be proportional to the – usually large – number of solutions produced and would aggravate the already tedious task of manual solution evaluation. On the contrary, during the user's solution evaluation both modules act *transparently*. In particular, the Decision Support Component also evaluates solutions and compares the performance of the user's Decision Support profile with his/her actual choices. Moreover, at the same time, the Machine Learning Component learns user preferences based exclusively on user's evaluation of the solutions. The Decision Support Component is available from the very first scene the user submits as an alternative means of *automatic solution evaluation*. If the user desires, he/she may assign solution evaluation entirely to the Decision Support Component.

If the user decides to continue with manual evaluation, after a number of sessions, i.e. scenes submitted and evaluated by the user, the Machine Learning Component – given that the user is consistent with a basic set of preferences – outperforms the decision support component and/or is able to adequately generalise thus foreseeing user's evaluation of newly generated solutions. At this point the user has two choices: to continue manual solution evaluation, thus further improving the accuracy of the Machine Learning Component, or to assign *automatic solution evaluation* entirely to the Machine Learning Component.

From the above it becomes apparent that based on regular user input and minimal overhead that is required only once for each user and is of acceptable complexity, our system offers two alternative methods for automatic solution evaluation:

- One based on the Decision Support Component, which is available from the very first use of the system and
- One based on the Machine Learning Component, which is available after a number of sessions of regular system use on behalf of the user.

7.2.3 Single User Profile for Diverse Declarative Descriptions

The proposed and implemented Intelligent User Profile Module offers a mechanism able to acquire and maintain a user profile, modelling a specific user's preferences, which is applicable to all previous and future declarative descriptions submitted by the user. In other words, the preference model obtained for a specific user is not restricted to only one of his/her declarative descriptions but applies, in a sense, *horizontally* to all past and future declarative descriptions submitted by the same user.

7.2.4 Reusability of User Profile and Generated Solutions

Following from the previous, the stored user profile may subsequently be applied not only to solutions generated based on the specific user's declarative descriptions but also to solutions already generated and stored in the Open-MultiCAD database, possibly for the sake of another user. In particular, the user may review and approve stored *declarative descriptions* for which solution generation has already taken place. Next, the user is able to request solution evaluation for the specific scene based on his/her personal user profile. As a result, the repetition of solution generation, which is an expensive task in terms of time, is avoided and the solutions already generated are evaluated for an alternative user.

Nevertheless, apart from creating a new declarative description, the user is always able to review an already stored description and modify it in order to generate solutions according to his/her custom variation. However, the ability to equally apply the user's profile to newly or already generated solutions, combined with the reusability of the solutions – produced for the sake of one user but subsequently evaluated by many – contributes to an overall increase in efficiency without sacrificing the flexibility of the system.

7.3 Future Work

Closing the current stage of this work a number of interesting directions have become apparent for further research and experimentation.

One of them stems from the idea to replace the pre-defined set of observed attributes with attributes isolated through the application of feature extraction techniques at the geometric representation of the produced and/or human evaluated solutions. It is worth noting that such an approach could lead to a custom set of observed features, possibly different for each user, which, in turn, would require re-engineering of the corresponding Machine Learning and Decision Support Components of the Intelligent User Profile Module. Another issue to consider in this case is the level of detail that will serve as the basis of the available information. In particular, feature extraction could be based on pure geometric information, i.e. dimensions and positions of objects, or a more informed interpretation of each solution representation, taking into account the interpretation of declarative relations and properties appearing in the corresponding declarative description.

Another interesting direction to explore would be the effort to apply the user preference knowledge, which is currently obtained during the solution evaluation stage, to the solution generation stage. This could prevent the generation of uninteresting solutions for the specific user, thus resulting to the obvious advantage of faster system response. The main disadvantage of such an approach would be the potential elimination of interesting solutions during the solution generation stage due to incomplete training or expected error of the corresponding intelligent mechanism. Moreover, the solution generation module of the existing environment would have to be modified in order to provide the appropriate interface for the current form of the user preference model.

Apart from the aforementioned directions, that form the main axes of future research, a few more could be explored separately or in parallel to them. In particular, it would be interesting to apply alternative learning algorithms and/or decision support techniques and experiment with already existing user preference information in the current environment. This experimentation could imply additional modules for the already existing system implementing these mechanisms.

Last but not least, the paradigm of declarative modelling of buildings could be replaced by the declarative modelling of other types of objects, not necessarily physical. Examples of the latter case could include the abstract definition of policies, system organisations, etc. and the representation of their inherent constraints and relationships through the Declarative Modelling methodology. User preferences in this context could represent executive decisions and choices with respect to alternative courses of action or the layout of the functional units respectively. Such an approach could broaden the scope of the Declarative Modelling methodology and the corresponding Intelligent User Profile Module presented in this work, leading to an assessment of their potential application to an alternative category of problems.

Conclusions

Dans ce travail nous avons combiné et adapté les concepts provenant des domaines *d'aide à la décision multi-critères* et *d'apprentissage automatique* et nous les avons appliqués au contexte des méthodologies de *la modélisation déclarative*. Notre but a été d'enrichir un environnement de conception déclarative déjà existant par des dispositifs intelligents qui améliorent la réponse de cet environnement aux préférences de chaque utilisateur. L'échange des idées entre les domaines mentionnés ci-dessus ainsi que l'adaptation et l'application des concepts spécifiques au prototype qui a été établi, ont fourni une nouvelle perception et des résultats fructueux dans le domaine de la modélisation déclarative. Par ailleurs, notre travail a donné des résultats intéressants concernant la flexibilité des techniques utilisées d'analyse de décision et d'apprentissage automatique et aussi concernant le degré d'adaptabilité qu'ils offrent lorsqu'ils sont transférés à un autre domaine.

Comparaison aux autres travaux

Malgré l'éventail de travaux concernant l'acquisition des préférences d'utilisateur et son application dans des domaines variés, peu seulement ont été proposés dans le domaine de la modélisation déclarative. En particulier, [Plemenos02] représente un tel effort dans ce contexte, tandis que [Champciaux98a] aborde le même problème dans un environnement très similaire. Néanmoins, comme il est expliqué en détail dans cette section, tous les deux travaux fonctionnent sous un ensemble différent d'hypothèses rapportant de ce fait une forme différente de modélisation des préférences d'utilisateur.

Dans la proposition [Plemenos02], deux approches alternatives ont été implémentées pour modeler les préférences d'utilisateur. Selon la première, la structure de chaque scène déclarative soumise est associée à un réseau neuronal dédié. Ce réseau est subséquemment entraîné avec les scènes que l'utilisateur approuve afin d'acquérir l'interprétation par l'utilisateur des relations et des propriétés déclaratives contenues dans la description spécifique. Après la période d'entraînement, le réseau peut évaluer et présenter à l'utilisateur seulement les scènes qui sont conformes à ses préférences. De même que dans notre approche, le mécanisme proposé ne réduit pas le nombre de solutions générées par le système puisqu'il traite les représentations géométriques produites par le générateur des solutions. En fait, les auteurs mentionnent comme future extension la possibilité d'appliquer l'évaluation du réseau neuronal *pendant* la génération des solutions, en empêchant un certain nombre de solutions d'être exploré par le mécanisme de satisfaction de contrainte exécutant la génération des solutions. Jusqu'ici, cette idée n'a pas été explorée plus loin.

La différence principale du Module Intelligent de Profil d'Utilisateur proposé et mis en application dans ce travail en comparaison avec la proposition mentionnée ci-dessus provient de la nature *orientée à description* du mécanisme d'apprentissage automatique utilisé. En particulier, la méthodologie mentionnée ci-dessus acquiert et maintient de manière précise des préférences d'utilisateur séparément pour chaque description déclarative. Dans la pratique, après un certain nombre de sessions, l'utilisateur aura soumis un certain nombre de descriptions déclaratives diverses, chacune représentée par son propre réseau neuronal, qui sera capable d'identifier les solutions *approuvées par l'utilisateur* pour la description correspondante. Cependant, cette configuration implique que, dès qu'une nouvelle description sera soumise par l'utilisateur, la connaissance précédemment acquise concernant ses préférences ne peut pas être exploitée sur les solutions nouvellement produites. En d'autres termes, la configuration proposée peut *généraliser* sa connaissance seulement dans les limites des descriptions déclaratives *précédemment vues*. En revanche, la construction du Module Intelligent de Profil d'Utilisateur proposé dans ce travail, rend la connaissance précédemment acquise entièrement exploitable afin d'évaluer des solutions provenant d'une nouvelle description déclarative. Naturellement, même dans notre cas, et comme il est devenu évident pendant les expériences, la *généralisation n'est pas toujours réussie* puisque chaque description peut indiquer des aspects différents des préférences de l'utilisateur. Cependant, dans un nombre considérable d'expériences, notre mécanisme a pu suffisamment généraliser, en choisissant avec succès les solutions approuvées par l'utilisateur parmi un ensemble de nouvelles solutions proposées. Ceci est également dû à l'hypothèse fondamentale que nous avons assumée que chaque utilisateur est intéressé à un ensemble spécifique d'attributs observés, qui, successivement, offrent un certain degré de *normalisation* parmi des solutions produites par des descriptions différentes. Dans un sens, le mécanisme proposé dans [Plemenos02] aborde le problème de préférences d'utilisateur *verticalement,* dans les limites d'une seule description déclarative chaque fois, alors que nous avons abordé le même problème *horizontalement,* à travers l'intervalle complet des descriptions déclaratives passées et futures potentiellement soumises par l'utilisateur.

La méthodologie alternative proposée à [Plemenos02] est fondée sur des techniques génétiques. En particulier, une première population de solutions est produite et graduellement améliorée par les opérations génétiques standards de croisement, mutation et clonage. L'évaluation des solutions par l'utilisateur après l'inspection joue le rôle de la fonction d'évaluation exigée dans le contexte des techniques génétiques. La différence principale de cette approche est qu'elle sert principalement comme un mécanisme de génération de *recherche de solution* se fondant sur la rétroaction répétitive de l'utilisateur. Contrairement au mécanisme précédent, ne sont pas produites toutes les solutions conformes à la description déclarative d'entrée. Ceci implique une réaction plus rapide du système comme avantage en compensation à l'exigence d'une intervention considérable de l'utilisateur et à la possibilité que certaines (probablement meilleures) solutions peuvent ne jamais être produites – à cause de la nature localisée de recherche des techniques génétiques. En revanche, le Module Intelligent de Profil d'Utilisateur proposé et mis en application dans ce travail, exige une intervention minimale de l'utilisateur et garantit la découverte des toutes des solutions intéressantes au prix d'une réaction globale du système plus lente. Néanmoins, la comparaison entre les deux approches n'est pas entièrement valide puisque l'une représente une technique de *recherche de solution* tandis que l'autre fonctionne en mode d'*exploration*, toutes les deux étant discutées dans le chapitre de la Modélisation Déclarative. En fait, les avantages et les inconvénients mentionnés plus haut représentent le résultat des interprétations alternatives impliquées par les deux modes de modélisation déclarative.

L'approche proposée à [Champciaux98a] comporte deux modules principaux : un classifieur non supervisé, assignant des solutions correspondant à chaque description déclarative à une hiérarchie de classe automatiquement produite fondée sur la similitude, à l'aide de l'algorithme Cobweb [Fisher87] à cette fin, et d'un module d'interface d'utilisateur qui permet à l'utilisateur d'examiner et évaluer les représentants de classe, entraînant de ce fait une mécanisme d'apprentissage automatique supervisé pour identifier un concept cible, c'est-à-dire une interprétation plus restrictive de la description initiale. Le mécanisme repose sur l'hypothèse que les solutions semblables partagent un degré similaire de préférences de l'utilisateur. Par ailleurs, le dernier des modules mentionnés plus haut implique l'exécution à une mode plus proche à la *recherche de solution* dans le sens qu'on élimine des classes entières des solutions semblables par l'évaluation de l'utilisateur pour rapporter un petit sous-ensemble de solutions géométriquement semblables représentant le concept cible. Enfin et surtout, la hiérarchie de concepts et le concept cible définis par l'utilisateur sont définis sur la base d'une seule description déclarative chaque fois, indiquant ainsi une considération *verticale* des préférences de l'utilisateur. En considérant la notion du concept cible similaire à celui des préférences d'utilisateur, le Module Intelligent de Profil d'Utilisateur proposé et mis en application dans ce travail fournit une flexibilité bien plus grande dans leur définition, dans le sens que l'utilisateur est libre de suggérer des évaluations différentes pour les solutions autrement similaires par le Composant d'Apprentissage Automatique tandis que le Composant d'Aide à la Décision peut toujours être utilisé pour classifier des solutions similaires. De nouveau, la différence la plus importante est qu'un modèle commun aux préférences de l'utilisateur, celui qui est acquis et mis à jour par l'évaluation des descriptions déclaratives alternatives, peut être immédiatement appliqué à toutes les solutions nouvellement générées, provenant d'une description déclarative précédemment inconnue. En d'autres termes, les deux approches servent un objectif différent, celle de [Champciaux98a] visant à raffiner la description déclarative suggérée par la classification et la rétroaction d'utilisateur, tandis que celle présentée dans ce travail vise à acquérir un modèle global des préférences d'un utilisateur particulier par l'utilisation régulière du système.

Les avantages pour la modélisation déclarative

La raison principale de ce travail provient du problème de l'explosion combinatoire qui est inhérente à la modélisation déclarative. Les descriptions simples contenant quelques objets et relations entre eux, peuvent conduire à centaines de milliers de solutions valides alternatives. Les capacités du matériel informatique moderne permettent la production de toutes les solutions alternatives dans une fraction du temps qui, autrefois, était nécessaire pour la même tâche. Cependant, la capacité des gens de passer en revue ces solutions et d'appliquer leur jugement afin de choisir celles qui sont plus proches de leurs préférences reste plus ou moins la même au cours des années. C'est l'une des raisons pour laquelle nous avons placé la tâche de l'évaluation intelligente de solutions au centre de notre recherche.

Ensemble de solutions amélioré

La modélisation déclarative offre à l'utilisateur la capacité de commencer par une description abstraite de haut niveau d'un objet, dans notre contexte une scène de bâtiment, le dispensant de la nécessité d'indiquer ses propriétés géométriques concrètes. Par ailleurs, le fait que la description d'entrée est abstraite et, jusqu'à un certain degré, ambigüe peut mener à des solutions alternatives qui peuvent intéresser l'utilisateur, même si celles-ci n'étaient pas initialement conçues par lui comme faisant partie des résultats.

L'incorporation du Module Intelligent de Profil d'Utilisateur à l'environnement Open-MultiCAD offre à l'utilisateur l'avantage de recevoir un sous-ensemble de la population de solutions qui est plus près de ses préférences personnelles. L'inspection de toutes les solutions produites est encore possible à la discrétion de l'utilisateur puisque le module présente les solutions produites par ordre décroissant de préférence. De cette manière, l'utilisateur qui n'est pas disposé à examiner toutes les solutions produites peut choisir d'arrêter le passage en revue des solutions après quelques représentants, sachant que les solutions qu'il a vues sont les plus proches de ses préférences. Ce fait représente réellement une approche alternative au mode de *recherche de solution*, discutée dans le chapitre Modélisation Déclarative, pour l'environnement Open-MultiCAD. L'utilisateur est, en effet, présenté, d'une façon transparente, avec le(s) meilleur(s) représentant(s) de l'espace de solutions correspondant à la description déclarative de la scène soumise comme entrée. Ceci correspond exactement à la notion du mode de recherche de solution et, dans le contexte actuel, la qualité des solutions est reliée aux préférences de l'utilisateur spécifique.

Surcoût minimal pour le système

L'essence de modélisation déclarative est la capacité de l'utilisateur de décrire une scène sous une forme abstraite évitant, de ce fait, les détails géométriques concrets. Conformément à ceci, nous avons respecté, dans tout ce travail, cette forme abstraite d'entrée puisqu'elle constitue un élément fondamental de la modélisation déclarative. Par conséquent, nous avons évité de demander une quelconque entrée supplémentaire concernant chaque scène et nous avons choisi de concentrer l'opération du mécanisme transparent sur la représentation géométrique des solutions.

La seule entrée supplémentaire exigée de l'utilisateur comporte une tâche de *prétraitement*, qui a lieu *une fois pour chaque utilisateur*, où il répond à un ensemble de questions concernant ses préférences afin d'initialiser le *profil d'aide à la décision* correspondant. Les questions portent sur l'importance des caractéristiques observées et n'exigent pas d'information géométrique explicite.

Après cette initialisation, l'utilisateur procède à l'utilisation régulière du système, soumettant des descriptions déclaratives et évaluant des solutions comme il l'aurait fait normalement. Même à cette étape, notre soin principal a été d'éviter l'introduction de surcoût supplémentaire pour l'utilisateur en ce qui concerne l'évaluation de solution. La raison est qu'un tel surcoût serait proportionnel au nombre – habituellement grand – de solutions produites et aggraverait la tâche déjà pénible de l'évaluation manuelle de solutions. Au contraire, pendant l'évaluation de la solution par l'utilisateur, les deux modules agissent *d'une manière transparente*. En particulier, le Composant d'Aide à la Décision évalue également des solutions et compare l'exécution du profil d'aide à la décision de l'utilisateur à ses choix réels. Par ailleurs, en même temps, le Composant d'Apprentissage Automatique apprend les préférences de l'utilisateur basé exclusivement sur l'évaluation de solutions de l'utilisateur. Le Composant d'Aide à la Décision est disponible à partir de la première scène que l'utilisateur soumet, en tant que moyen alternatif d'*évaluation automatique de solutions*. Si l'utilisateur le désire, il peut confier l'évaluation de solutions entièrement au Composant d'Aide à la Décision.

Si l'utilisateur décide de continuer l'évaluation manuelle, après un certain nombre de sessions, c'est-à-dire de scènes soumises et évaluées par l'utilisateur, le Composant d'Apprentissage Automatique – en supposant que les choix de l'utilisateur sont cohérents avec un ensemble de préférences de base – surpasse le Composant d'Aide à la Décision et il peut généraliser suffisamment, prévoyant, de ce fait, l'évaluation par l'utilisateur des solutions nouvellement produites. En ce moment l'utilisateur a deux choix : continuer l'évaluation manuelle de solutions, améliorant la précision du Composant d'Apprentissage Automatique, ou confier entièrement *l'évaluation automatique de solutions* au Composant d'Apprentissage Automatique.

D'après ce qui précède, il devient évident que, basé sur l'intervention normale de l'utilisateur plus un surcoût minimal, qui est exigé seulement une fois pour chaque utilisateur et est de complexité acceptable, notre système offre deux méthodes alternatives pour l'évaluation automatique de solutions :

- Une méthode basée sur le Composant d'Aide à la Décision, qui est disponible même à la première utilisation du système et
- Une méthode basée sur le Composant d'Apprentissage Automatique, qui est disponible après un certain nombre de sessions d'utilisation régulière du système par l'utilisateur.

Profil d'utilisateur unique pour des descriptions déclaratives diverses

Le Module Intelligent de Profil d'Utilisateur proposé et implémenté offre un mécanisme capable d'acquérir et mettre à jour un profil d'utilisateur, modelant les préférences de chaque utilisateur, qui est applicable à toutes les descriptions déclaratives précédentes et futures soumises par l'utilisateur. En d'autres termes, le modèle de préférence obtenu pour un utilisateur particulier n'est pas limité seulement à une de ses descriptions déclaratives mais s'applique, dans un sens, *horizontalement* à toutes les descriptions déclaratives, passées ou futures, soumises par le même utilisateur.

Réutilisabilité du profil d'utilisateur et des solutions produites

Suivant ce qui précède, le profil d'utilisateur enregistré peut subséquemment être appliqué non seulement aux solutions produites basées sur les descriptions déclaratives de l'utilisateur spécifique mais, également, aux solutions déjà produites et enregistrées dans la base de données d'Open-MultiCAD probablement pour un autre utilisateur. En particulier, l'utilisateur peut passer en revue et approuver les *descriptions déclaratives* enregistrées pour lesquelles la génération de solutions a déjà eu lieu. Ensuite, l'utilisateur peut demander l'évaluation de solutions pour la scène spécifique basée sur son profil d'utilisateur personnel. En conséquence, la répétition de la génération de solution, qui est une tâche chère en termes de temps, est évitée et les solutions déjà produites sont évaluées pour un autre utilisateur.

Néanmoins, en dehors de la création d'une nouvelle description déclarative, l'utilisateur peut toujours examiner une description déjà enregistrée et la modifier afin de produire des solutions selon sa variation individuelle. Cependant, la capacité d'appliquer également le profil d’utilisateur aux solutions nouvelles ou déjà produites, combinée avec la réutilisabilité des solutions – produites pour un utilisateur mais subséquemment évaluées par plusieurs – contribue à une augmentation globale d’efficacité sans y sacrifier la flexibilité du système.

Recherche future

En terminant ce travail, un certain nombre de directions intéressantes sont devenues évidentes pour pousser plus loin la recherche et l’expérimentation.

L’une d’elles provient de l’idée de remplacer l’ensemble prédéfini d’attributs observés par des attributs isolés à travers l’application de techniques d’extraction de caractéristiques sur la représentation géométrique des solutions qui sont produites et/ou évaluées par une personne. Il vaut la peine de noter qu’une telle approche pourrait mener à un ensemble personnalisé de dispositifs observés, probablement différent pour chaque utilisateur, qui aurait, à son tour, besoin de la reconfiguration des composants correspondants d’Aide à la Décision et d’Apprentissage Automatique du Module Intelligent de Profil d’Utilisateur. Un autre point à considérer dans ce cas-ci est le niveau du détail qui servira comme base de l’information disponible. En particulier, l’extraction de caractéristiques pourrait être basée sur l’information géométrique pure, c’est-à-dire les dimensions et les positions des objets, ou sur une interprétation mieux renseignée de chaque représentation de solution, tenant compte de l’interprétation des relations déclaratives et des propriétés apparaissant dans la description déclarative correspondante.

Une autre direction intéressante à explorer serait d’essayer d’appliquer la connaissance des préférences de l’utilisateur, qui est actuellement obtenue pendant l’étape d’évaluation de solutions, à l’étape de génération de solutions. Ceci pourrait empêcher la génération des solutions inintéressantes pour l’utilisateur spécifique, résultant, de ce fait, à l’avantage évident d’une réaction de système plus rapide. L’inconvénient principal d’une telle approche serait l’élimination potentielle de solutions intéressantes pendant l’étape de génération de solutions due à un entrainement incomplet ou à l’erreur prévue du mécanisme intelligent correspondant. D’ailleurs, le module de génération de solutions de l’environnement existant devrait être modifié afin de fournir l'interface appropriée pour la forme actuelle du modèle de préférences de l'utilisateur.

A part les directions mentionnées ci-dessus, qui forment les axes principaux d'une recherche future, quelques autres pourraient être explorées séparément ou en parallèle avec elles. En particulier, il serait intéressant d'appliquer des algorithmes alternatifs d'apprentissage automatique et/ou des techniques d'aide à la décision et les expérimenter avec l'information de préférences d'utilisateurs déjà existante dans l'environnement actuel. Cette expérimentation pourrait impliquer des modules additionnels pour le système déjà existant, implémentant ces mécanismes nouveaux.

Enfin, le paradigme de modélisation déclarative de bâtiments pourrait être remplacé par la modélisation déclarative d'autres types d'objets, pas nécessairement physiques. Quelques exemples du dernier cas pourraient inclure la définition abstraite de politiques, d'organisations de systèmes, etc. et la représentation de leurs contraintes et rapports inhérents par la méthodologie de la Modélisation Déclarative. Les préférences d'utilisateur dans ce contexte pourraient représenter les décisions exécutives et les choix en ce qui concerne les lignes d'action alternatives ou la disposition des unités fonctionnelles respectivement. Une telle approche pourrait élargir la portée de la méthodologie de Modélisation Déclarative et du Module Intelligent de Profil d'Utilisateur présenté dans ce travail, menant à une évaluation de leur application potentielle dans une autre catégorie de problèmes.

References

[Baldi01] Baldi P., Brunak S., Bioinformatics: The Machine Learning Approach, MIT Press, 2001.

[Bardis04] Bardis G., Miaoulis G., Plemenos D., An Intelligent User Profile Module for Solution Selection Support in the Context of the MultiCAD Project. 7e Infographie Interactive et Intelligence Artificielle (3IA) International Conference, Limoges, France, 2004.

[Bardis05] Bardis G., Miaoulis G., Plemenos D., Learning User Preferences at the Declarative and Geometric Description Level. Short Paper. 8e Infographie Interactive et Intelligence Artificielle (3IA) International Conference, Limoges, France, 2005.

[Bardis06] Bardis G., Miaoulis G., Plemenos D., Intelligent Solution Evaluation Based on Alternative User Profiles, Enterprise Information Systems VII, pp. 103-111, Springer, 2006.

[Barral00] Barral P., Dorme G., Plemenos D., Visual understanding of a scene by automatic movement of a camera. Short paper. Eurographics 2000.

[Bonnefoi02] Bonnefoi P.-F., Plemenos D., Constraint Satisfaction Techniques for Declarative Scene Modeling by Hierarchical Decomposition, 3IA, Limoges, France, 2002.

[Bonnefoi04] Bonnefoi P.-F., Plemenos D., Ruchaud W., in alphabetical order, Declarative Modelling in Computer Graphics: Current Results and Future Issues, ICCS 2004, LNCS 3039, pp. 80–89, 2004.

[Borges92] Borges J.L., Labyrinths, Kastaniotis Editions, 1992 (in Greek), originally in *El jardin de senteros que se bifurcan, 1941.*

[Boser92] Boser B., Guyon I., Vapnik V., A training algorithm for optimal margin classifiers. Fifth Annual Workshop on Computational Learning Theory. ACM Press, Pittsburgh, 1992.

[Boughanem94] Boughanem M., Plemenos D. Techniques d'apprentissage en modélisation déclarative de scènes par décomposition hiérarchique. Infographie Interactive et Intelligence Artificielle 3IA'94, Limoges, 1994.

[Brans85] J.P. Brans, P. Vincke, A Preference Ranking Organisation Method: The PROMETHEE Method for MCDM. Management Science, 31(6):647–656, 1985.

[Buckley85] Buckley, J.J. (1985). Fuzzy Hierarchical Analysis, Fuzzy Sets and Systems, 17(3):233-247, 1986.

[Burbidge01] Burbidge R., Buxton B., An Introduction to Support Vector Machines for Data Mining, Keynote, 12th Young Operation Research Conference, 2001.

[Champciaux98a] Champciaux L., Classification: a basis for understanding tools in declarative modelling, Computer Networks and ISDN Systems 30, 1841–1852, 1998.

[Champciaux98b] Champciaux L. Gestion des contraintes géométriques pour l'aide à l'aménagement urbain. PhD thesis, Ecole des Mines de Nantes, 1998.

[Chauvat94] Chauvat D., The VoluFormes Project: An Example of Declarative Modelling with Spatial Control, PhD Thesis, Nantes, France, 1994.

[Deerwester90] Deerwester S., Dumais S. T., Landauer T. K., Furnas G. W., Harshman R. A., Indexing by Latent Semantic Analysis, Journal of the Society for Information Science, 41(6), 391-407, 1990.

[Doherty92] P. Doherty, D. Driankov, and A. Tsoukiàs. Partial logics and partial preferences. In Proceedings of the International Conference on Economics, Management and Information Technology CEMIT'92, Tokyo, pages 525–528, 1992.

[Dragonas05] Dragonas J., Makris D., Lazaridis A., Miaoulis G., Plemenos D., Implementation of Collaborative Environment in MultiCAD Declarative Modelling System, 8^{e} Infographie Interactive et Intelligence Artificielle (3IA) International Conference, Limoges, France, 2005.

[Dubarle89] D. Dubarle. Essai sur la généralisation naturelle de la logique usuelle. Mathématique, Informatique, Sciences Humaines, 107:17–73, 1989.

[Elman93] Elman J.L., Learning and Development in Neural Networks: The Importance of Starting Small, Cognition 48, pp. 71-99, 1993.

[ExpertChoice05] http://www.expertchoice.com/productsservices/ourapproach.htm

[Fisher87] Fisher D., Knowledge Acquisition Via Incremental Conceptual Clustering, Machine Learning 2: 139-172, 1987.

[Freund97] Freund Y., Schapire R., A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, August 1997.

[Gaildrat03] Gaildrat V., Modélisation déclarative d'environnements virtuels: Création de scènes et de formes complexes par l'énoncé de propriétés et l'emploi d'interactions gestuelles, Habilitation à Diriger des Recherches, L'Université Paul Sabatier (Toulouse III), France, 2003.

[Gaildrat05] Gaildrat V., Modélisation déclarative d'environnements virtuels – Tour d'horizon, Équipe Synthèse d'Images et Réalité Virtuelle, IRIT – Toulouse III, 2005.

[Golfinopoulos05] Golfinopoulos V., Miaoulis G., Plemenos D., A Semantic Approach for Understanding and Manipulating Scenes, 8^{e} Infographie Interactive et Intelligence Artificielle (3IA) International Conference, Limoges, France, 2005.

[Goldberg89] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Corporation Inc., 1989.

[Goodwin04] Goodwin P., Wright G., Decision Analysis for Management Judgement, Third Edition, Wiley, 2004.

[Holland92] Holland, J. H. Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, 1992.

[Hornik89] Hornik K., Stinchcombe M., White H., Multilayer Feed-forward Networks Are Universal Approximators, Neural Networks, Volume 2, Issue 5, Pages 359-366, 1989.

[Howard66] R.A. Howard, Decision analysis: Applied decision theory, Proceedings of the Fourth International Conference on Operational Research, pages 55-71, Wiley-Interscience, 1966.

[Jackson86] Jackson, P., Introduction to Expert Systems, Addison-Wesley, Reading, 1986.

[Kechman01] Kechman V., Learning and Soft Computing – Support Vector Machines, Neural Networks and Fuzzy Logic Models, MIT Press, 2001.

[Keeny76] R.L. Keeny and H. Raiffa, Decisions with Multiple Objectives: Preference and Value Trade-offs, J. Wiley & Sons, New York, 1976.

[Konnar00] Konnar A., Artificial Intelligence and Soft Computing – Behavioral and Cognitive Modeling of the Human Brain, CRC Press, 2000.

[Leshno93] Leshno M., Lin Ya.V., Pinkus A., Schocken S., Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function, Neural Networks volume 6, number 6, pp. 861-867, 1993.

[Lewitt03] Lewitt M., Polikar R., An Ensemble Approach for Data Fusion with Learn++, Multiple Classifier Systems, 4th International Workshop, MCS 2003, Guilford, UK, June 11-13, 2003, Lecture Notes in Computer Science 2709, Springer 2003.

[Lucas89] M. Lucas, D. Martin, P. Martin, D. Plemenos, The ExploFormes project: Some Steps Towards Declarative Modelling of Forms. AFCET-GROPLAN Conference, Strasbourg (France), November 29 – December 1, 1989. Published in BIGRE, no 67, pp 35-49 (in French).

[Makris03] 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, Infographie Interactive et Intelligence Artificielle (3IA) International Conference, Limoges, France, 2003.

[Makris05] Makris D., Architectural Styles in Declarative Modelling, PhD Thesis, Université de Limoges, France, 2005.

[Martin99] Martin D., Martin P., PolyFormes: Software for the Declarative Modelling of Polyhedra, The Visual Computer (1999) 15:55-76, Springer-Verlag, 1999.

[McCarthy95] McCarthy J., What Has AI in Common with Philosophy? Aaron Sloman's Symposium on philosophy and AI, IJCAI-95, 1995.

[MCDA-SAS05] Multiple Criteria Decision Analysis – State of the Art Surveys, Springer 2005.

[Miaoulis02] 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, Doctoral Thesis (in French), Université de Limoges, France, 2002.

[Mitchell98] Mitchell M., An introduction to Genetic Algorithms. MIT Press, Cambridge, Mass, 1998.

[Papadimitriou00] Papadimitriou C., Raghavan P., Tamaki H., Vempala S., Latent Semantic Indexing: A Probabilistic Analysis, Journal of Computer and System Sciences 61, pp.217-235, 2000.

[Plemenos91] Pléménos D., Contribution à l'étude et au développement de techniques de modélisation, génération et visualisation de scènes: le projet MultiFormes, Professorial Dissertation, Nantes, 1991.

[Plemenos95] Plemenos D., Declarative modeling by hierarchical decomposition – The actual state of the MultiFormes project. GraphiCon'95, St Petersbourg, Russia, July 3-7, 1995.

[Plemenos02] Plemenos D., Miaoulis G., Vassilas N., Machine Learning for a General Purpose Declarative Scene Modeller, International Conference GraphiCon, Nizhny Novgorod, Russia, 2002.

[Polikar01] Polikar R., Udpa L., Udpa S.S., Honavar V., Learn++: An Incremental Learning Algorithm for Supervised Neural Networks, IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 31, No. 4, November 2001.

[Polikar02] Polikar R., Byorick J., Krause S., Marino A., Moreton M., Learn++: A Classifier Independent Incremental Learning Algorithm, Proceedings of Int. Joint Conf. Neural Networks, pp. 1742—1747, 2002.

[Poulet94] Poulet F., Modélisation déclarative de scènes tridimensionnelles: Le projet SpatioFormes, Infographie Interactive et Intelligence Artificielle (3IA), Limoges, 1994.

[Poulet96] Poulet F., Lucas M., Modelling Megalithic Sites, EuroGraphics, Volume 15, Issue 3, pp.279-288, 1996.

[Ravani03] Ravani I., Makris D., Miaoulis G., Constantinides P., Petridis A., Plemenos D., Implementation of Architecture-oriented Knowledge Framework in MultiCAD Declarative Scene Modeling System, 1st Balcan Conference in Informatics, Thessaloniki, Greece, 2003.

[Roberts02] Roberts R., Goodwin P., Weight Approximations in Multi-attribute Decision Models, Journal of Multicriteria Decision Analysis 11: 291-303 2002.

[Roy68] B. Roy. Classement et choix en présence de points de vue multiples (la méthode ELECTRE). RIRO, 8:57–75, 1968.

[Ruchaud02] Ruchaud W., Plemenos D., MultiFormes: A Declarative Modeller as a 3D Scene Sketching Tool, International Conference on Computer Vision and Graphics (ICCVG), Zakopane, 2002.

[Russel02] Russel S., Norwig P., Artificial Intelligence – A Modern Approach, Second Edition, Prentice-Hall, 2002.

[Saaty80] T.L. Saaty, The Analytic Hierarchy Process. New York: MacGraw-Hill, 1980.

[Sanchez00] Sanchez S., Le Roux O., Gaildrat V., Luga H., Résolution d'un problème d'aménagement spatial à l'aide d'un algorithme génétique, AFIG'00, IMAG Grenoble, pp 113–122, 2000.

[Schapire90] Schapire R.E., The strength of weak learnability. Machine Learning, 5(2):197–227, 1990.

[Sellinger97] Sellinger D., Plemenos D., Interactive Generative Geometric Modelling by Geometric to Declarative Representation Conversion, WSCG'97, pp. 504-513, 1997.

[Sellinger98] Sellinger D., La modélisation géométrique déclarative interactive. Le couplage d'un modeleur déclaratif et d'un modeleur classique. Thèse de doctorat, Limoges, 1998

[SemanticLight06] http://www.semanticlight.com/

[Takane77] Takane Y., Young F. W., de Leeuw J., Psychometrika 42. 7-67, 1977.

[Turing50] Turing, A.M. 1950. Computing Machinery and Intelligence, From Mind LIX, no. 2236 : 433-460, Oct. 1950.

[Vapnik68] Vapnik V.N., Chervonenkis A.Y., On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. Doklady Akudemii Nuuk USSR 181 (4) (in Russian), 1968.

[Vapnik98] Vapnik V.N., Statistical Learning Theory, Wiley, 1998.

[Vassilas02] Vassilas N., 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, SETN 203-214, Thessaloniki, Greece, 2002.

[Vincke92] Vincke P., Multicriteria Decision-aid, Wiley, 1992.

[Witten05] Witten I.H., Frank E., Data Mining – Practical Machine Learning Tools and Techniques, 2nd Ed., Elsevier, 2005.

[Xu02] Xu K., Stewart J., Fiume E., Constraint-based Automatic Placement for Scene Composition, Graphics Interface, Canada, 2002.