Architecture and the Built environment

#05
2012

CItyMaker

Designing Grammars for Urban Design

José Nuno Beirão

CItyMaker

Designing Grammars for Urban Design

José Nuno Beirão
Delft University of Technology, Faculty of Architecture, Department Urbanism
Department Architectural Engineering + Technology

CItyMaker

Designing Grammars for Urban Design

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op October 16th, 2012 om 12.30 uur
door JOSÉ NUNO DINIS CABRAL BEIRÃO
architect
geboren te Torres Novas, Portugal

Dit proefschrift is goedgekeurd door de promotoren:
Prof. Sevil Sariyildiz
Prof. Henco Bekkering

Samenstelling promotiecommissie:

Rector Magnificus, Voorzitter
Prof.dr.ir. Sevil Sariyildiz, Technische Universiteit Delft, promotor
Prof.ir. Henco Bekkering, Technische Universiteit Delft, promotor
Dr.ir. Rudi Stouffs, Technische Universiteit Delft, co-promotor
Prof.dr. José Pinto Duarte, Technical University Lissabon
Prof.dr. Juval Portugali, Tel Aviv University
Prof. Vicent Nadin, Technische Universiteit Delft
Assoc.prof. Dr. Stephen Marshall, U-College London
Dr. Stephen Read, Technische Universiteit Delft
Prof.dr.ir. V.J. Meyer, Technische Universiteit Delft, Reservelid

Prof.dr. José Pinto Duarte, heeft als begeleider in belangrijke mate aan de totstandkoming van het proefschrift bijgedragen.

The author is a member of the City Induction research project funded by the Fundação para a Ciência e Tecnologia (FCT) under Grant PTDC/AUR/64384/2006 and hosted by the ICIST at TU Lisbon. José Nuno Beirão was funded by the FCT under Grant SFRH/BD/39034/2007. The City Induction research was coordinated by Prof. José Pinto Duarte.

Architecture and the Built environment

abe.tudelft.nl

Ontwerp: Sirene Ontwerpers, Rotterdam
ISBN 978-1479355020
ISSN 2212-3202

'To Elsa for her endless patience and to my parents for their continuous yes '

Table of contents (concise)

Foreword

This thesis is aimed at two different types of readers: those interested in shape grammars and those interested in urban design. The former will find extensive and detailed concepts that address (urban) design synthesis using shape grammars for design exploration. The latter will find the conceptual basis for the development of generative design tools for urban design. It encompasses a design method using design tools based on shape and description grammars as the generative formalism. The research was developed within the framework of a research project called City Induction (Duarte et al., 2012). The goal of the project is the integration of design support tools to formulate, generate and evaluate urban designs. This thesis focuses on the development of generation tools for urban design, corresponding to the more design-oriented component of City Induction. It defines the theoretical model for an urban design tool called CItyMaker and presents two prototype implementations as proof of concept. As such, urban designers will find the main contributions of this thesis in the discussion of design methods and conceptual tools to support an efficient, flexible, interactive, and responsive urban design process. Although integrated into the research, all the methods and tools developed in this thesis were designed to be autonomous, that is, to work and be used independently of the other City Induction modules.

For those interested in shape grammars the thesis may represent a step forward towards a more extensive use of the generative properties of shape grammars in urban design. In particular, a method has been developed to implement shape grammar-based codes embedding a conventional reflective design method. From the point of view of the urban designer, the thesis will provide a design method for urban design and a supporting generative tool with an interactive and responsive working environment. The approach shown here is responsible for creating an environment which enhances awareness of the effects of design decisions throughout a progressively evolving urban design process.

José Nuno Beirão, Delft, May 2012

Acknowledgements

I would like to thank José Duarte for his continuous and invaluable support. My research work is undoubtedly indebted to him.
I would like to thank my supervisors, Sevil Sariyildiz and Henco Bekkering, for their belief in me and support and also my co-supervisor, Rudi Stouffs who has offered me continuous support since my first day at TU Delft. Above all, I will remember their friendship.
I would like to thank Jorge Gil and Nuno Montenegro for their involvement in the work. Much of the work done during my PhD research involves their direct collaboration in the context of the City Induction research project. The City Induction project has been funded by the Fundação para a Ciência e Tecnologia (FCT), Portugal (PTDC/AUR/64384/2006), hosted by the ICIST at TU Lisbon, and coordinated by José P. Duarte. Working with Jorge, José and Nuno was a highly stimulating experience.
I also wish to thank:
Pirouz Nourian, for proposing the development of the parametric model and for his involvement in it – his support has been of the utmost importance to my work;
Cândido Chuva Gomes, Frits Palmboom and Frits van Dongen for kindly providing documentation for their plans and for giving up their time to answer my questions and comments; Ekim Tan for the first push and for involving me in participatory urban design; Eng. Assis Lopes, for his technical advice; Pedro Arrobas, for his enthusiasm in continuing the development of the parametric model; Frank van der Hoeven, for his initial support and alternative views; Horácio Ramos for his hospitality and challenging discussions.
I must also thank my closest colleagues for the discussions we have had that certainly have influenced and transformed my convictions as a researcher: Michela Turrin, André Chaszar, Bige Tunçer, Antje Kunze, Irem Erbas, Olgu Çaliskan, Bardia Mashhoodi, Alper Alkan, Gabriela Celani, Ozer Ciftcioglu and Michael Bittermann.
Finally, I would like to thank the FCT (Fundação para a Ciência e Tecnologia, Portugal) for financially supporting my PhD - grant SFRH/BD/39034/2007.

Summary

Due to its complexity, the evolution of cities is something that is difficult to predict and planning new developments for cities is therefore a difficult task. This complexity can be identified on two levels: on a micro level, it emerges from the multiple relations between the many components and actors in cities, whereas on a macro level it stems from the geographical, social and economic relations between cities. However, many of these relations can be measured.

The design of plans for cities can only be improved if designers are able to address measurements of some of the relationships between the components of cities during the design process. These measurements are called urban indicators. By calculating such measurements, designers can grasp the meaning of the changes being proposed, not just as simple alternative layouts, but also in terms of the changes in indicators adding a qualitative perception.

This thesis presents a method and a set of tools to generate alternative solutions for an urban context. The method proposes the use of a combined set of design patterns encoding typical design moves used by urban designers. The combination of patterns generates different layouts which can be adjusted by manipulating several parameters in relation to updated urban indicators. The patterns were developed from observation of typical urban design procedures, first encoded as discursive grammars and later translated into parametric design patterns. The CItyMaker method and tools allows the designer to compose a design solution from a set of programmatic premises and fine-tune it by pulling parameters whilst checking the changes in urban indicators. These tools improve the designer's awareness of the consequences of their design moves.

Samenvatting

Vanwege haar complexiteit, is de evolutie van steden moeilijk te voorspellen. Het plannen van nieuwe ontwikkelingen voor steden is dan ook een moeilijke taak. Deze complexiteit kan worden bevestigd op twee niveaus: op microniveau volgt de complexiteit uit de vele relaties tussen de vele componenten en actoren van de steden; op macroniveau volgt de complexiteit uit de geografische, sociale en economische betrekkingen tussen de steden. Echter, veel van die relaties kunnen worden gemeten. Het ontwerpen van plannen voor steden kan alleen worden verbeterd als ontwerpers in staat zijn om tijdens het ontwerpproces metingen over een aantal van de relaties tussen de componenten van steden aan te pakken. Deze metingen worden stedelijke indicatoren genoemd. Door deze metingen uit te voeren, kunnen ontwerpers de betekenis van de veranderingen die worden voorgesteld begrijpen, niet alleen als eenvoudige alternatieve indelingen, maar ook ten aanzien van de wijzigingen in de indicatoren.
Dit proefschrift toont een methode en een reeks van tools om alternatieve oplossingen voor een stedelijke context te genereren. De methode stelt het gebruik voor van een combinatorische verzameling van ontwerppatronen die typische ontwerp-zetten die door stedenbouwkundigen gebruikt worden coderen. De combinatie van patronen genereert verschillende indelingen die kunnen worden afgestemd door het manipuleren van een aantal parameters in confrontatie met bijgewerkte stedelijke indicatoren. De patronen werden ontwikkeld na observatie van typische stedenbouwkundige procedures, eerst gecodeerd als discursieve grammatica's en vervolgens vertaald in parametrische ontwerppatronen. De CItyMaker methode en instrumenten laten een ontwerper toe een ontwerp-oplossing samen te stellen uit een reeks van programmatische vooronderstellingen en af te stemmen door het bewerkstelligen van de parameters tijdens het controleren van veranderingen in de stedelijke indicatoren. Deze instrumenten verbeteren de bewustwording van ontwerpers over de gevolgen van hun ontwerp-zetten.

Table of contents (extended)

1 Introduction

This section provides an overview of the research topic addressed in this thesis. It starts with an introduction to the reasons for designing for the complexity of cities, presenting the idea of developing new tools and methods for designing flexible urban systems as a response to the problem of complexity. It then explains the context of the research, namely its role as part of a larger research project and provides a brief overview of the tools and methods proposed in the thesis to address the problem.

§ 1.1 Flexibility in dealing with the complexity of cities

Cities are dynamic systems. Their configuration, spatial and social characteristics are the result of a large number of factors that are in some way involved in, or influence, the way a settlement is built, as well as its dynamics. Many of these factors are difficult to control and some may be highlighted due to their relative importance in most urban contexts: topography; geographic location; climate; economic dynamics; social dynamics.

The emergence of settlements and the growth of cities seem to be directly related to the economy of places (Jacobs, 1970). If the economic dynamics of a place is growing, the settlements supporting this economy are likely to grow too. During the 20th century such growth achieved proportions never experienced before by mankind and the beginning of the 21st century shows an even faster growth dynamic.

What was in the past, especially in the Middle Ages, the common growth pattern, namely local self-organised growth defining what is usually known as an organic grid, can no longer accommodate the growth needs of the present day, at least with the minimum comfort and environmental conditions considered normal by current standards. The unfortunate examples of organic growth that we can see around the world involve extremely poor living conditions. Extensive areas of informal growth can be found, especially in developing countries, known as slums, *favelas*, *barrios* or other terms depending on the cultural context. Although lively places, in most cases these slums offer very degrading living conditions which quite clearly stress the need for developing new planning strategies (Acioly Jr., 2010).

The need to respond to very extreme growth demands is also still valid in some developed countries, as is evident in the recently implemented Vinex programme in the case of the Netherlands (Boeijenga, Mensink, and Grootens, 2008). China is a very

specific case in which essentially top-down solutions are being applied. Whatever the context and the dichotomy between emerging (bottom-up) growth and planned top-down growth, it is desirable to control the urban spaces resulting from this growth, as a large amount of research on the subject seems to suggest (Provoost, 2010). Moreover, something may be learned from both processes and used creatively in new designs. One thing is clear: the tradition of designing urban space by producing authoritarian layouts is no longer an efficient strategy. Several authors point towards the need for planning flexible solutions using flexible processes. Incrementality, flexibility, adaptability, individuality and freedom of choice have become the key words used to express new approaches and strategies for urban planning and design (Correa, 2000).

At the end of his book Ascher (2001) identified the following as the new principles for urbanism:

- urban planning involving mechanisms for negotiating and elaborating solutions instead of designing layouts;
- reflexive urban planning, involving constant analysis;
- informed urbanism, prepared for the demands of sustainable development;
- participatory and flexible urbanism based on consensus;
- heterogeneous urban planning composed of hybrid solutions and stylistic openness.

Ascher suggests very generic approaches to urban design or planning which are radically different from the usual development of fixed layouts and certainly a departure from the typical aesthetically designed city of the modern era. Ascher also proposes negotiation as opposed to the traditional production of a fixed design. The traditional ways of planning and designing new urban areas do not respond efficiently to current needs. Contrary to the critical tendency directed towards planning methods, the majority of existing tools are still destined for the design of urban plans in the traditional layout. In all cases the aim is to produce a fixed layout for top-down implementation in a particular area. The object of the design is considered an isolated object instead of a complex, open system, which is what cities are (Portugali, 2009).

§ 1.2 Flexibility and flexible design

The term flexible design and, in particular, flexible urban design, although widely used by architects and urban designers (Gausa, Hammond, and Hammond, 1998), lacks a formal definition. Flexibility is a term used in engineering systems design to refer to the ability of a production system to deal with uncertainty (Gupta and Goyal, 1989).

The term is used in many texts on architecture and urban design, especially those associated with the design of housing systems (Gausa and Salazar, 2002) (Bosma, Van Hoogstraten, and Vos, 2000). Whatever the specific form and context of the term, the concept of flexibility and flexible design features in the architects' lexicon. Certain other approaches, although not directly addressing or defining this term, clearly use the concept as their main driving force. Some important publications clearly show this (e.g.: "A Pattern Language" (Alexander et al., 1977); "Supports" (Habraken, 1972); "Design for change" (Friedman 1997); shape grammar-based architectural research (Stiny and Mitchell, 1978) (Koning and Eizenberg, 1981); "Game Urbanism" (Venhuizen, 2010); urban grammars (Beirão and Duarte, 2005)). The common ground in all these approaches is the search for design systems which are able to deal with uncertainty and change in terms of problem definition. Most of them deal with rules and methods for providing systems of solutions rather than one single solution. A few approaches propose very detailed algorithmic formalisms to deal with the problem. Alexander et al's Pattern Language, for instance, provides a generic algorithmic structure open to interpretation. Shape grammars (Stiny and Gips, 1972) provide rigorous algorithmic formalisms to generate variations on a design language (Koning and Eizenberg, 1981).

Considering the above, the term *flexible design* can be defined as:

- **a set of design solutions for a specific design problem formulation, expressed through a specific set of design rules instead of the traditional fixed formal solution.**

This particular definition for *flexible urban design* refers exclusively to flexible design for urban planning. A flexible design therefore does not have a definitive shape until it is concluded and should not have one even on completion. A flexible design should respond to variations in needs. On an urban scale, however, flexibility can be addressed on two main levels: *design flexibility* and the *flexibility of the design*. The former refers to the capacity of the design method or process to adapt to changes in the problem formulation and the latter to the fact that a specific final design is still capable of accommodating change and evolving during and after implementation – that is, it refers to its adaptability over time. Flexible design, as defined above, represents a third level of flexibility.
This thesis addresses the production of design tools and methods for *flexible urban design* considering both *design flexibility* and the *flexibility of the design*. The tools and methods are destined to be used by designers but the tool structure and methods are defined in ways that support easy visual interactivity in a collaborative environment involving stakeholder participation. In the thesis the term "designer" refers both to a single designer and a design team, the latter being likely to represent the most common situation in the case of urban design.

Flexible design is a complex approach, as it should be used in many possible contexts involving many different strategies in order to propose a plan. Any kind of design task is developed by a process of negotiation between problem formulation and solution, making use of analysis, synthesis and evaluation (Lawson, 2006). In order to produce a design (whether flexible or not), the designer needs to carry out several analytical tasks, generate several solutions and evaluate several possible solutions before reaching a definitive one. The process is not sequential. The designer is likely to reformulate the problem several times when confronted with an unwanted design evolution or unwanted solutions. This thesis concentrates on the process of synthesis, attempting to understand the rules underlying the design moves (Schön, 1987) that progressively compose urban designs, in order to reuse them to generate new designs.

§ 1.3 Tools and methods for urban design

To date, several new planning processes and tools have been developed and implemented with the aim of improving the quality of the areas planned. In order to achieve this, urban designers have come up with two basic lines of action:

1 Implementing changes to the traditional urban design process;

2 Developing tools to support urban designers and improve the quality of designs.

The first line of action involves changes in design practice, such as:

1 Integrating all actors involved in the city development decision-making process by introducing participatory methods into standard procedures. (Arnstein, 1969) (Kunze and Schmitt, 2010) (Tan, 2009).

2 Promoting diversity by subdividing the process into partial areas to be developed by different design teams collaboratively. This process can be subdivided many times on different levels of scale e.g. (Venema, 2000) (de Maar, 1999) and is already common practice in the Netherlands.

3 Designing basic guidelines and generic rules, leaving local decisions to the stakeholders involved. (Habraken, 1980) (Friedman, 1997) (Beirão and Duarte, 2009).

4 Designing with patterns (Alexander et al., 1977) (Salingaros, 2000) or codes (Carmona, Marshall, and Stevens, 2006) to support design decisions. Rather than

being simply prescriptive, this idea is based on the principle that certain recurrent design problems provide efficient and already tested solutions that can subsequently be reapplied.

This is not an extensive list but these approaches cover the most common strategies for changing the design process. With regard to the second line of action, many different kinds of tools have been developed to support urban design, which essentially fall into two main categories:

1 tools used for enhancing information about the way cities grow and the processes involved in their growth, which are not directly involved in the design process, and
2 tools used directly to improve the design practice.

The tools in both categories can be said to be design support tools but only those in the second category are also design tools. The former are essentially analysis and simulation tools.
Some approaches use analytical methods to improve the quality of information on the nature of cities and why certain phenomena occur in them. These are strictly analytical tools. Some common approaches focus on the behaviour of urban space, taking its topological structure or the topological structure of the street network into consideration, and these include space syntax (Hillier and Hanson, 1984), place syntax (Stahle, Marcus, and Karlström, 2005), and route structure analysis (Marshall, 2005).
The use of simulation processes can enhance awareness of phenomena that may influence the evolution of certain urban contexts and provide insights into how alternative solutions may evolve over time or according to specific changing conditions. For instance, cellular automata have been used to define simulation models to explain urban sprawl, understanding the way it spreads and eventually predicting future expected developments (Batty, 2005). Both Batty (2005) and Portugali (2000) have dedicated extensive studies to understanding the complex behaviour of cities. However, Portugali cites the non-linear behaviour of cities to explain why certain urban phenomena cannot be predicted.
Other approaches try to replicate the real conditions of unpredictability and multi-agent participation by setting gaming environments to simulate these conditions, allowing several people to participate in city games. City games are set to replicate the main rules and conditions of a real-case scenario (Mayer et al., 2009) (Venhuizen, 2010). "Serious game" strategies represent an ambiguous approach that lies between design and simulation. In principle, the process is a simulation but, depending on the way the games are set and the context in which they are set, they may eventually end up with real propositions or simply supporting real propositions. "Serious games" focus on managing the growth process rather than design. They can also be used as negotiation platforms in which different stakeholders are the players. The gamming concept, or serious games as it is commonly called, may use computer game environments, but the concept can also be implemented with handmade game

environments. The important thing is really how the game is defined in terms of the required interaction between the participants and the goals of the game. Examples can be found in (Venhuizen, 2010), (Mayer et al., 2009) and (Tan, 2009) [WS1].

The book "Model Town" (Stolk and Brömmelstroet, 2009) shows, in ten chapters, different alternative models for urban simulation, urban design and their use in new town planning. Several approaches are explored in the book: models based on fractals and cellular automata presented to understand and describe urban sprawl (Batty 2009); studies on self-organisation in the development of cities using cellular automata and agent-based models (Portugali, 2009); studies on urban dynamics using agent-based models (Timmermans, 2009); game-based models for urban simulation (Mayer et al., 2009); pattern and rule-based design approaches (Beirão and Duarte, 2009) and cellular automata (König, 2009) for designing new towns; space syntax for understanding the failures of English new towns (Karimi et al., 2009); design based on interactive negotiation platforms (Lehnerer, 2009); and studies on urban sprawl based on geographic information systems analysis (Bonfiglioli, Calza, and Stabilini, 2009). Except for the approaches in Chapters 7 (König, 2009), 8 (Lehnerer, 2009) and 9 (Beirão and Duarte, 2009), all the other chapters essentially focus on understanding the complexity of city developments rather than designing cities. Some models presented in the book have been developed to enhance knowledge of urban behaviour, and others to directly support or inform design decisions or the design process. The book still stands as a good survey of the available tools and methods for urban design simulation.

However important simulation tools and techniques may be in informing urban design, urban simulation should be regarded as having different goals to urban design. Design aims at reshaping or transforming the world by proposing a new state of things, envisaged as solving a problem by improving the existing conditions of a context in the initial state of the design (Cross, 2007). Analytical and simulation approaches may produce information that will enhance the designer's awareness of the initial state of a design and its context, and also the consequences of design decisions, comparing them with known standards. In this sense, one important aim would appear to be integrating analytical methods and tools with design methods and tools. Simulation can be seen as a design support method, but not necessarily as design.

Density indicators are some of the most commonly used devices to inform, analyse or establish goals in urban design. It is fairly common practice to develop plans by designing a layout for specific density goals and it is also common to establish the negotiation process with stakeholders based on a layout and the respective density indicators. The important role of urban indicators in urban design and planning was addressed in detail by Berghauser-Pont and Haupt in their book Spacematrix (2010). The first part of the book provides an extensive survey of the role of density measures in urban planning and design, the kind of indicators commonly used, their meanings and inconsistencies, and how designers have used and still use them. Beirão (2005) also produced a survey of the role of urban indicators in urban design and planning in the Portuguese context, where similar practices regarding the role of density measures and

indicators were encountered. The important thing to stress here is that density measures and other urban indicators play a very important role in the urban design process. They can be seen as constraints or design goals by either designers or stakeholders, and they can also be seen as planning or controlling devices by municipal planners or planners in general. In all cases, testing designs against density indicators is common practice on the part of all the actors involved in the urban design process and this is used, albeit for different purposes, at different levels of scale.
Whatever tools are used for designing urban plans, a proposed urban morphology always needs to be tested against density measures and other analytical data, since in most circumstances the design goals are somehow expressed in this way. Furthermore, through density measures some of the qualities of the urban fabric can be understood, as the second half of Berghauser-Pont and Haupt's book shows. However, the degree of complexity and unpredictability of the city implies the need for more design tools and methods. If we really want to learn how to design successful urban spaces, we also need tools and theories that define and evaluate what successful urban spaces are.

§ 1.4 The research context and main goals

This thesis addresses a specific part of a larger research project called City Induction (Duarte et al., 2012).
As defined, City Induction [WS2] aimed to develop an urban design tool by integrating Computer Aided Design (CAD) into a Geographic Information System (GIS) environment. The main idea was to take advantage of the existing tools for urban analysis and use them directly in conjunction with generative design tools for design synthesis. The City Induction project aims to formulate urban design briefs from an analysis of contextual data, generating design alternatives by following the design brief and evaluating the design against the specifications of the brief. To this end, the project includes three models that support the whole concept: (1) a model for formulating the specifications of the urban programme, taking the available data on the context into consideration (the formulation model – 4CityPlan); (2) a model for generating alternative urban designs based on shape grammars (the generation model – CItyMaker); and (3) a model for evaluating the designs in terms of sustainability and design performance (the evaluation model – EvModule). Nuno Montenegro and Jorge Gil are responsible for developing the formulation model and the evaluation model respectively. José Duarte is responsible for project coordination. This thesis concerns the development of the generation model – CItyMaker.

The theories supporting each model defined in the research project can be described as follows:

- the formulation model bases the analytical functions on GIS assessment tools and a knowledge base – an ontology (Gruber, 1993) – from which a detailed description of programme specifications is defined as programming patterns (Alexander et al., 1977) and formalised as description grammars (Stiny, 1981);
- the generation model bases its generative qualities on the development of compound forms of description and shape grammars, also following pattern-like structures;
- the evaluation model is responsible for the validation of the designs, using spatial and network analysis tools such as space syntax (Hillier and Hanson, 1984) to perform the assessment.

The conceptual framework was based on Duarte's concept of discursive grammars (2001). Independently of the larger research project, the focus of this thesis lies in the development of an urban design tool. Urban design is addressed in this thesis as having the following characteristics:

- Urban design is a collaborative decision-making practice involving the transformation of territories from rural or urban to upgraded urbanised forms, taking sustainability into account[1].
- Urban design bases its decisions on territorial analysis, including context analysis, and on data derived from candidate design solutions. This should represent fundamental information for participatory decision-making.
- The role of the urban designer or urban design team is to introduce expert vision and skills into the decision – making process.
- The result of an urban design process is a system of solutions in formats that allow for easy visual assessment, supported by adequate tools.
- The process of designing an urban system should be as interactive as possible, allowing for a bidirectional flow between problem formulation and solution.

1 The term sustainability is used superficially and irresponsibly nowadays. It is not the goal of this thesis to address sustainability issues and define what this means in terms of the development of cities. However, it is expected that certain main driving forces behind sustainable urban design may be addressed in the methods and tools proposed in the thesis. The main area of reference for this is directly related to programme formulation and can be found in (Montenegro 2010)). In any case, the expression *taking sustainability into account* as used above means an uncompromising commitment to considering the incorporation of the means of addressing basic sustainable urban design criteria without claiming that the tool aims to design sustainable plans. Without involving building design, such a claim would be fraudulent.

The thesis proposes a generative formalism for developing compound forms of spatial grammars for urban design synthesis encompassing the reflective characteristics of the design process (Schön, 1983). These grammars were called Urban Grammars. In order to allow for a correct approach to their use, a supporting design method for urban design is proposed, based on an interactive rule-based approach. This theoretical framework defines urban grammars in a progressive fashion by gradually adding small grammars, each generating typical urban design instructions, or moves, to use Schön's term. These small sets of design instructions are defined as design patterns (Gamma et al., 1995) for urban design. Technically, these design patterns are algorithms composed of discursive grammars encapsulating the behaviour of the design move. These generative design moves were called Urban Induction Patterns (UIPs). This thesis shows that urban designs are obtained from a full formulation-generation-evaluation cycle of UIPs, each of which is also a limited formulation-generation-evaluation cycle corresponding to a typical urban design move. It also shows that this structure is compatible with the reflective responsiveness that characterises design practice. This responsiveness is maintained by allowing the designer to interact in the generative process through interfaces which provide options and extensive parameter manipulation. The consequences of each design move are assessed by means of a geometrical model and data generated and expressed in terms of urban indicators and other properties of the urban fabric.

Following this theoretical framework, the thesis presents two partial implementations of an urban design generation tool – CItyMaker. CItyMaker is defined as an autonomous urban design tool that can be used independently of the formulation and evaluation models being developed for the City Induction research project. In order to be used for design, the generation system (like any other design tool) should be used following a negotiation between problem and solution involving analysis, synthesis and evaluation. The design tool proposed in this thesis provides a means for entering the analytical data obtained by any means and methods available and also provides tools for outputting design solutions and the data derived in terms of urban density indicators and other derived properties, thus allowing for immediate visual evaluation of the solutions. The evaluation can be enhanced with any other available evaluation tools, namely GIS, space syntax, place syntax or tools being developed within the context of the City Induction evaluation model. In all cases, the designs are generated in formats that allow for their immediate insertion into a GIS environment.

§ 1.5 Contributions

The main scientific contributions of the thesis are:

1 A theoretical model for an urban design tool involving generative design capabilities and a design method for its use. The theoretical model provides a structure for urban design generation compatible with a GIS representational structure, including calculations for density based indicators. The model provides a flexible design platform for the production of flexible urban designs. The flexibility space is defined by a specific urban grammar that is synthesised during the design process. This topic contributes to the field of computational methods applied to urban design theory.

2 An ontology describing the concepts involved in the urban design process, contributing to the development of knowledge bases for urban design.

3 A shape grammar formalism for developing urban grammars, contributing to the field of shape grammar studies.

4 A set of recommendations for developing software for urban design and city information modelling (CIM) software, with regard to how it should be structured to support GIS interoperability, thus contributing to the field of computational methods applied to urban design theory.

5 A design method to enhance the quality of the information flow supporting design decisions in an urban design process, contributing to the field of computational methods applied to urban design theory and to urban design practice. The method enables design decisions to be based on better grounded information.

6 A tool for supporting studies on the relationship between urban morphology and density, contributing towards improving designer awareness of such relations and eventually to devising more accurate links between these measures and the quality of urban space.

The contributions of this knowledge to design practice are likely to improve the quality of urban design, its management and its response to complexity. In other words, the abovementioned contributions will allow for improvements to flexibility in the urban design process on three levels: flexibility during the design process; the production of flexible designs, i.e. systems of solutions; greater adaptability in design. These qualities should, according to current theories, improve the quality of new urban developments, especially in terms of how they respond to the complexity of evolution. Without adding any other meaning to the term sustainability than the internationally accepted one, the new approaches proposed in this thesis will certainly represent a step forward towards the production of more sustainable cities, at least in the sense that they provide a greater capacity to design cities that are capable of adapting to evolving societies.
The improvement in data flow during the design process is also likely to improve the efficiency of participatory processes, in the sense that the proposed systems will allow for alternative scenarios to be considered and supporting data to be provided for each scenario. The different stakeholders will therefore be able to evaluate decisions better, based on the improved information on the alternative scenarios.

2 Problem definition and research questions

The research problem addressed in this thesis can be summarised as finding appropriate tools and methods to design efficient, flexible urban plans as a response to the problem of designing for the complex behaviour of cities. It addresses flexible urban design, considering both *design flexibility* and the *flexibility of the design* (see definitions in Chapter 1 and Appendix 1). In order to develop such design tools a proposal is presented based on compound forms of shape and description grammars. Shape grammars (Stiny and Gips, 1972) and description grammars (Stiny, 1981) are generative formalisms used to generate designs in a recursive fashion. The idea is to develop systems of solutions rather than one single solution, thus allowing the system to adapt to changes in the environment.
However, specific technical problems emerge from the use of shape grammars regarding the construction of semantically appropriate designs. Therefore, this thesis will also present research into the use of shape grammars for designing.

§ 2.1 The generic problem – the complexity of cities

The main problem stemming from complexity theories of cities is that cities evolve in ways that are difficult to predict and that city developments, even when planned, tend to find patterns of organisation that were not previously defined in the plans. Portugali (2000) explains this behaviour by stating that cities are non-linear open systems subject to the behaviour of several other open systems. For instance, human social behaviour is also a complex system which directly interferes with the way cities evolve. Interactions between the systems occur both at global and local levels. Portugali gives one example (2009), namely a global housing policy in Israel some years ago, which started with the prediction of a growth need in housing due to a large influx of Russian immigrants, and triggered a local reaction in the population who started renting rooms and houses, usually in better locations. This resulted in a much lower need for new housing than initially expected, creating a major financial failure out of what had seemed to be a secure investment. The problem, Portugali explains, is that prediction is itself part of the complex system and influences the way the system behaves. Similarly, any designs for an urban system become part of the system, influencing the

final result. At any given moment a city is the result of global interactions, through top-down decisions, and local interactions through bottom-up individual decisions.
At this point a question emerges:

How can we plan cities if their behaviour is so complex and unexpected? [1]
This question clearly identifies the generic problem addressed in this thesis. Several authors have indicated various answers, presenting different strategies and viewpoints to address the problem. The strategies can be summarised as follows:

1 Simulation techniques – some authors have proposed several different simulation techniques as a way of predicting future scenarios, but as we have seen in Portugali's example this can be a misleading strategy. Typical examples include cellular automata techniques for simulating urban sprawl (Batty, 2005) or city games applied to specific contexts (Mayer et al., 2009). Despite Portugali's remarks, these techniques allow us to foresee eventual future scenarios which may help the designer to reach better informed decisions. It is the role of the expert to judge the reliability of the predictions, their meanings and the extent to which such information will influence the outcome.
2 The use of systems, patterns and types – some authors stress the value of historically accumulated knowledge, tradition or tacit knowledge which, with regard to architecture and cities, are values expressed or embedded in systems, patterns and types. Types are recurrent solutions bearing evidence of practical success from repeated use through time. Types offer an underlying social agreement on ways of living, building and behaving in society. Patterns have similar qualities, but they relate a problem occurrence in the environment to a typical solution supported by empirical, tacit, or scientific evidence. They are *recipes* for solving recurrent problems occurring in the environment. A system is generic; it embeds a set of constructive solutions that allow for a large amount of freedom in composition. A system defines constructive solutions and does not imply specific spatial organisation. Examples of an architectural system are the classical or the international style. (Alexander, 1964), (Habraken, 1988), (Alexander et al., 1977), (Habraken, 2000).
3 The use of evaluation techniques – evaluation techniques propose theories and tools for analysing design propositions in particular contexts and comparing them with acceptable standards for validation. Evaluation techniques, however, have two main problems. First they can only be performed after the design is finished and therefore do not help with the design decision-making process since they only either validate it or not. Secondly, the standards used for comparing the results also need to be validated. In this sense they are dependent on selection criteria, i.e. the definition of a system of values which can, in itself, be a research problem. However, resorting to accepted and well established types and patterns to select such criteria may be an efficient approach to defining valid quality standards for the purposes of comparison. Also, some criteria are essentially objective, for instance, determining,

whether something consumes energy or not. In such cases, a value can be measured. Nevertheless, the tangible meaning of these values, especially in relation to the context, is always a matter for interpretation (Gil and Duarte, 2008). Most evaluation techniques are supported by GIS analytical methods or spatial analysis, such as space syntax (Hillier, 1996) or place syntax (Stahle, Marcus, and Karlström, 2007). The large number of evaluation studies using GIS and space syntax clearly shows the potential of these techniques but also stresses the need for expert interpretation of the analytical results obtained from GIS or space syntax analysis.

4 Participatory decision-making – the main driving force behind participatory decision-making is democratic principles (Arnstein, 1969) but it is also a response to the complexity problem in the sense that it brings the agents of complexity into the decision-making process. The agents of complexity are those (individuals or collectives) who can locally or globally influence urban environment developments. Participatory decision-making can also be supported by different methods and support tools used to inform or improve the quality of such decisions. Simulation interfaces and gaming interfaces are probably some of the most common strategies (Stolk and Brömmelstroet, 2009). Games can be set as participatory events in many different ways. Nowadays several different alternative methods and strategies can be found for implementing city games (Tan, 2009) (Venhuizen, 2010). City games have to be adapted to the specific design problems being studied and should be defined in ways that simulate real conditions. They should also be goal oriented in the sense that if the output of the game can be considered close to a real design proposition, then the game becomes more than a simulation and turns into a design product. The game itself can also be set as an implementation process and, as such, can become a managing tool. The role of the computer tools in supporting this kind of approach still offers vast potential for exploring the interaction of different agents with the urban environment. Tools and methods are usually adapted to the subject in question, following specific procedures (Slocum, 2003). Furthermore, participatory decisions can be further improved if analytical support tools are used to inform the stakeholders participating in the events. However, the analytical results need to be interpreted and the interpretation can be aided by experts and visualisation tools.

5 Flexibility and flexible design – flexibility is cited by several authors as a design strategy for dealing with complexity. Ascher (2001) talks about developing an *urbanism of devices instead of designing plans* and Friedman (1997) talks about *designing for change*, defining a *development vision* or code for particular contexts. Flexibility in engineering systems is generically defined as the capacity of a system to produce different kinds of solutions. More specifically, there are two different but complementary definitions: (1) *the capability of a system to overcome known changes in the environment* and (2) *the capability of a system to cope with unpredictable changes* (Gupta and Goyal, 1989). These definitions are both interesting and extendible to design. The second definition, however, involves the difficulty of dealing with the unexpected. Three main kinds of interpretation of the

term "flexibility" in the context of designing can be identified. (1) The first considers the capacity of the design method or process to adapt to changes in the programme of requirements (*design flexibility*). (2) The second considers the design of systems of solutions rather than one single solution (multiplicity – the design of systems; *flexible design*). (3) The third considers the design of solutions which are capable of adapting even after the implementation is finished (*the flexibility of the design*). The ideal approaches should embrace all three kinds of flexibility simultaneously.

§ 2.2 Generative rules for design flexibility

Considering the 5 strategies explained above for dealing with the complexity of cities, flexibility seems the most complete because in a certain way it can include the others. If the three stated kinds of flexibility are considered in a design process, the system designed should be able to produce many different solutions and respond to unexpected local changes. In this sense such a system is capable of simulating many alternative scenarios. There is also no reason for not using types and patterns to design flexible systems and thus select certain qualities attributed to those types and patterns in order to incorporate them into the flexible design. Evaluation techniques can be used to support the selection of solutions and inform decision makers about the relative qualities of the available solutions. It can also offer added value in terms of participatory urban design by proposing systems of solutions with associated analytical data and evaluation results that the participants can discuss and decide upon. Considering the above, flexibility has been chosen in this thesis as the main approach for dealing with the generic research question [1] of designing for the complexity of cities.

A more specific question emerges from the former:

What tools and methods can be used to develop flexible urban design solutions, simultaneously embedding the five-point strategy for dealing with complexity? And what are the characteristics of such tools? [2]

Partial answers to this question are provided by previous work.
In urban design, the design process usually flows through three separate processes: analysis, synthesis and evaluation. The processes involve different experts and tools, and are different in terms of their nature and goals:

- Analysis involves geographic contextual analysis using GIS and aims to establish a consensual and clear view of the contextual requirements. Analysis may involve all kinds of experts.
- Synthesis aims at developing possible plans for future scenarios responding to the requirements identified using visualisations, plan representations and an output of related data. Representations and visualisations are usually produced by designers, typically using CAD software but also other supporting software.
- Evaluation aims to validate a selected solution, usually through plan representations inserted into the existing context again using a GIS environment in which the solutions can be tested and validated.

These characteristics of the urban design process are particularly important because they usually involve separate procedures and separate tools. The structure of the urban design process tends to adopt a linear strategy addressing the three activities in the stated order. However, design synthesis, according to Lawson (2006), incorporates the three activities in a design process and they are usually implemented in any order. Lawson explains the design activity as a problem and solution negotiation process using analysis, synthesis and evaluation, in which the order is not really mandatory but depends on the designer's personal methods. He states that the design process can actually begin with a hypothetical solution – a primary generator (Darke, 1979) – which is then evaluated in order to better inform the problem description. In conclusion, whatever the regular workflow of an urban design process may be, synthesis, i.e. the plan design, should always involve the three activities as much as possible. The better the analytical processes and the more integrated they are, the better the results of design synthesis will be in terms of urban design. Therefore, the integration of the analytical and synthesis tools should be improved, namely through better integration of the GIS and CAD tools in the urban design workflow. In particular, the importance of data flow – information support – should be stressed throughout the design process.

Urban design involves specific differences in comparison to conventional design processes, in particular architectural design or product design. It starts with an important difference: the object of the design is never a single object but a system of complex objects, each of which is the object of a design process and all of which involve complex functional, economic and symbolic relationships. Another important difference is that the object is also shaped by many different stakeholders including the final users and in principle the design decisions should be open to all of them[2]. What is argued here is that these differences add one particularly difficult characteristic to urban design – unpredictability.

2 Note that although a collective client or even a client association may be found in architectural design, this always involves a finite well-known number of actors, whereas in urban design this might be impossible to identify or predict in advance and may even change during the design process.

In a way the three activities - analysis, synthesis and evaluation - are a lot more complex in the case of urban design than in other design practices. Each activity will now be considered in detail.

Analysis

Essentially two analytical situations may be considered in relation to urban design: context analysis (prior to synthesis) and design analysis in context. In both situations the analytical process is complex and involves specific means and methods which are distinct from the typical design tools. For instance, analysis involves searching through large sets of data of many types (alphanumerical, vector and raster data). This information is sometimes stored in the databases of geographical information systems (GIS) and it addresses several kinds of data involving social, economic, and functional information complementing the vector or raster data. In simple terms, the analytical process essentially involves methods for pattern identification within the data, using data mining techniques, for instance (Witten and Frank, 2005), spatial analysis and other topology-based methods. The data is analysed by searching through the databases and the topological structure of the vector data. All these analytical processes involve very different tools from those typically used for design and usually supported by GIS.

Essentially, analytical tools are supported by GIS whilst design tools operate in CAD systems. These two kinds of tools have many interoperability problems both on a technical and a methodological level. Technically both the data and the representations have to follow particular topological structures and relations which link the representations to their meaning. On a methodological level, data and representations in GIS systems are not supposed to be altered but simply queried. In the synthesis process designers actually aim to change the existing conditions, i.e. generate representations of new transformations proposed for the existing reality, and CAD tools are the best tools to accommodate such a process. In this sense, analytical processes and synthesis processes are separate, not only because of their different nature but also due to the technical incompatibility of the systems. In practice, each time an urban design is produced the representation and corresponding data needs to be translated or adapted to fit the GIS structure if it is to be analysed in a context using the tools provided in the GIS platform. GIS practitioners spend a lot of time working on this translation procedure. The design workflow which should move easily between the three activities is therefore broken and separated. In practice most analytical activities are performed by expert teams separate from those in charge of synthesis activities, usually because GIS tools and CAD tools have totally different usage paradigms. The synthesis process is therefore supported mainly by visual assessment rather than an integrative analysis of each design move.

Synthesis

Synthesis is the generative process. It transforms the existing state of things into another state, solving a certain problem in the environment and satisfying certain previously defined goals. The synthesis process implies a profound knowledge of the conditions of the existing state (hence the need for analysis) and also the existence of design goals. The precision and detail of the goals is very much context-dependent. The definition of urban design goals is informed by several issues: those informed by the site and size of the intervention, by higher levels of urban regulations, by stakeholders, namely those who represent the investment power, by accepted sustainability indicators, standards or best practices and even those informed by hypothetical test solutions. Montenegro (2010) addresses strategies for the definition of urban design programmes in detail, identifying the main topics involved in the formulation of sustainable urban design briefs[3].

However independent the synthesis process is, the interoperability problem between GIS and CAD actually interferes negatively in the design process. The fact is that the existing data has an influence. The design decisions and, similarly, the measurements of the designs developed are needed for the evaluation tasks or simply for analysing the solutions integrated within the context. In fact, the evaluation activity is basically the result of a set of analyses of candidate solutions placed in context and confronted with a set of pre-validated standards.

Considering the above, it would appear evident that the whole design process would profit from better interoperability between GIS and CAD and that CAD tools would clearly support the synthesis process better if they were capable of reading data directly from the GIS database and generating new data relating to the designs generated in the GIS database in order to allow for further analysis in context. On the other hand, in product and architectural design the creative process essentially (and maybe desirably) lies in the hands of one designer or one decision-maker. In the case of urban design, decisions should be made by as many stakeholders as possible and therefore the normal characteristics of the decision-making process in design are subverted. Adding to these complications, stakeholders are sometimes interested in visualising a design proposal and alter it and at other times they are interested in visualising data or analytical results and altering them. These latter needs clearly show the advantage of combining analysis and synthesis by overcoming the CAD-GIS interoperability problems and allowing for simultaneous visualisation of solutions and data throughout the synthesis process.

3 The extent in which these goals are known at the beginning of the synthesis process depends on how much information is available as a result of the analytical procedures. However, goals, such as those informed by stakeholders, may change in the light of certain visualisations or intermediate proposals. This reinforces Lawson's view of design as a negotiation process.

Evaluation

Evaluation is based on analytical procedures but compares the results of these procedures with a previously validated system of values that is set as a benchmark. The problem regarding evaluation in urban design is that it is difficult to establish a globally valid system of values to use as benchmark. Most authors agree that urban design is very much dependent on contextual issues, whether morphological, geographical, cultural or social. Establishing urban design benchmarks is, in itself, a difficult problem which involves specific and localised research and participatory decision-making, not to mention sometimes conflicting economic interests. To add to these complications, benchmarks are also related to the programmatic goals of the design and, as already mentioned, problem formulation in design evolves through negotiation with the design solution. Thus the definition of benchmarks can become caught up in a closed circle. In any case, once an evaluation is made, it can be a great help for those involved in the design process to be able to visualise the results easily. Furthermore, a single solution or visualisation is often considered not conclusive enough and therefore the possibility of interactive models or multiple solutions is usually appreciated by the agents involved.

To conclude, due to the complexity of cities, and in terms of administrative procedures, several countries tend to separate these activities since involving several different teams in the decision-making process is seen as a positive practice. The reason for this is simple. This practice allows for democratic decisions involving shared responsibility and avoids assigning too much power to one decision-maker. However, this has fostered the separate development of tools for the three activities, creating specialised tools for the different activities. This is particularly evident in the case of England where it is common practice to subdivide decisions and specialised studies are frequently requested to support the decision-making process. When developed in this specialised format, the tools may sometimes be incompatible with the regular design practice workflow.

While designing, designers effectively approach their decision-making process by following a practice, as explained in Lawson's design model (2006). This means that as part of the design activity they analyse existing situations taking different kinds of geographical data into consideration, and gradually test different moves by defining incremental transformations for the existing conditions which are always confronted with some system of values or derived information that allows for an evaluation of the proposed transformations. Therefore it seems reasonable to say that any tools that aim to support design practice, even in urban design, need to provide as much access as possible to the available data and analytical procedures in order to support decisions regarding the existing conditions and how to transform them. Furthermore, in urban design, the designs need to be presented in ways that allow for a good understanding and visualisation of the proposed transformations for a given context but also maintain the flexibility needed to adapt to the various requirements of the stakeholders involved in decision-making.

Mayer et al. (2009), whilst supporting the use of serious games in urban planning and simulation, point out that tools and methods to support participation should be:

- Integrative, considering holistic and systemically different levels of design and decision-making.
- Dynamic, showing the performance of alternatives in relation to the behaviour and preferences of the participants.
- Interactive, supporting negotiation between stakeholders.
- Transparent, producing clear results which all participants can understand.
- Flexible, reusable and adaptable, therefore capable of adjusting to different design contexts and their subsequent evolution.
- Fast and easy to use.
- Communicative and educational, providing insight into the problem structure, alternatives and their particular perspectives.
- Authoritative, meeting analytical standards and political standards for validity in order to increase the possibilities of success.

At this point, the research question [2] could be reformulated as:

What is the structure of an urban design tool that is capable of generating flexible urban designs for a given context and providing data that improves understanding of the design? [3]

There is a technical question embedded in this:

How can we link GIS tools to CAD tools to obtain designs that contain some analytical results? [4]

Or in a simplified form:

How can we link analytical tools to design tools interactively? [5]

The main incompatibility problem between GIS analytical tools and CAD design tools is related to the topological structure of representations found in GIS environments. Most of the time consumed by GIS users is spent converting data into compatible standard formats. In most cases this task implies data transformation and in the specific case of representations (of a new plan, for instance), these need to be converted into a topologically correct structure thematically separating representations that consider theme and basic geometry simultaneously, i.e. in addition to thematic layering, the representations need to be separated into the basic geometric types, namely layers represented by points, layers represented by lines and layers represented by polygons.

The main concept proposed in this research is that if generative algorithms are used to generate design representations they can be programmed in such a way that the outputs fit into correctly structured representations. Shape grammars (Stiny and Gips, 1972), specifically compound forms of shape and description grammars (Stiny, 1981), can be used to define the generative algorithms needed for such a purpose. Shape grammars are a well known formalism for encoding the rules underlying design languages. They are both analytical and synthetic in the sense that they allow rules to be inferred in a language from a sample of designs in that language and also generate new designs within the language by applying the inferred rules. These rules are transformation rules that apply recursively to an initial shape to generate designs. A particular set of rules defines a shape grammar. A description grammar operates with design characteristics other than shape, for instance, name, use, street type, square type, material, or other. By using compound forms of these grammars, designs can be generated in a pre-defined structure, for instance, one which is compatible with the pre-requisites of GIS representation. The designs generated can therefore be easily integrated into the analytical procedures provided by the GIS platform.
This is the main computational formalism supporting the model for the urban design generator presented in this thesis. This formalism uses a set of parallel discursive grammars (Duarte, 2001) which are compound forms of shape and description grammars enhanced with some heuristics to reduce the computational resources. The discursive grammars define simple codes replicating urban design moves used recurrently by urban designers during the design process. The shapes and descriptions used by the grammars are representations of urban concepts expressing relationships previously defined in an ontology.

This thesis proposes the development of design support tools based on the abovementioned formalism. The contribution of this hypothetical tool to design practice is envisaged as:

- Providing the designer with the capacity to reflect on the morphological arrangements of the urban components needed for a plan, whilst providing analytical data that complements the understanding of such arrangements or, in other words, providing a means for confronting solutions with goal descriptions.
- Enhancing the designer's awareness of the meaning and consequences of their design decisions. This awareness is built up gradually throughout the design process.

On a technical level, shape grammars can provide a means of bridging the gap between GIS and CAD environments. However, there are technical difficulties relating to the development of shape grammar implementations which need to be addressed.

§ 2.3 Difficulties regarding the use of shape grammars

The difficulties regarding the implementation of shape grammars and shape grammar interpreters are well known. The points most frequently indicated as typical shape grammar problems basically fall into three categories:

1. matching parametric shapes;
2. providing the means for a correct semantic interpretation of shapes and defining meaningful designs;
3. defining grammars for design synthesis as a part of the design process without the constraints of using pre-defined languages.

These three problems have already been identified in specialist literature, but most solutions are partial, unsatisfactory or difficult to implement in practical terms.
Matching parametric shapes seems to be an ongoing unsolvable problem in shape grammar research, either because matching solutions cannot be found for every kind of shape or because shape grammar interpreters are rare and limited. Despite the large amount of interesting research into the development of algebras of shapes (Krishnamurti, 1980) (Stiny, 1991) (Stouffs, 1994), matching a shape can be subject to many kinds of ambiguities. In order to deal with such difficulties, several researchers have developed a substantial set of formalisms based on the mathematical foundations of shape grammars (Stiny, 1980) to tackle the matching problem (Krishnamurti and Earl, 1992) (Liew, 2003). Likewise, a vast collection of papers can be found on the uses of shape grammars in design analysis, which seems to be the most successful field in shape grammar research. Less work has been produced on their use in design synthesis and even less on the implementation of shape grammar interpreters.
Each new design problem addressed with shape grammars seems to need new formalisms or adjustments to supplement the existing ones. It seems that shape grammars will never be able to solve design problems without the need for new research. In addition, due to the matching problem, the implementation of shape grammar interpreters is still work in progress and there is no single shape grammar interpreter capable of implementing the shape grammar design potential in a satisfactory way.
The second problem, which will be termed the problem of semantics, deals with the meaning of shapes. The problem of semantics involves the various ways in which the meaning of shapes (or combined shapes) plays a role in seeing and deciding. Consider Figure 1, for example: if an urban designer was asked which rectangles represented buildings, which represented plots and which represented blocks, they would be able to answer immediately without any need for further information. However, in a shape grammar if the rule for adding a porch to the main façade applies to a building, as in Figure 2-a, how can we guarantee that the absurd design in Figure 2-b is not produced?

There are several ways of addressing this problem when developing a grammar. We can use labels for classifying shapes, parallel grammars with independent rules for each kind of representation, description grammars computing other meanings of shapes, colours, etc. However, even though these formalisms are already used to solve a great number of representation issues, a substantial number of design situations involving semantics still cannot be solved.

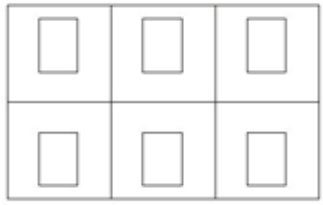


Figure 1
Urban block

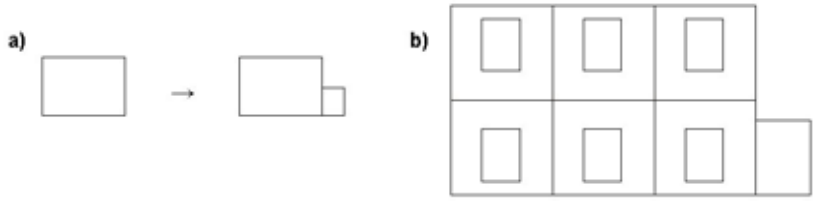


Figure 2
a) Rule for adding a porch to a building; b) Misinterpretation of the rule

However, the worst aspect of using shape grammars in a design synthesis process is that the design language needs to be known in order to apply a grammar, i.e. we need to know the rules in advance. Yet in any creative design process it is impossible to know the correct language to apply in advance. In fact, a design problem is likely to have many possible answers, each using different design languages depending on the design team involved in the process. This is the reason why competitions are common practice in architecture and urban design. A design language is not a premise of the design process; on the contrary, it is also one of the products of the design process. This problem is a consequence of the fact that shape grammars were inspired by the generative grammars of linguistics (Chomsky, 1957). Fleisher (1992) has already identified these flaws in shape grammars. Comparing them to natural languages, Fleisher points out that in natural languages the grammar is the set of syntactic rules that allows people to communicate in a common language. The syntactic rules are the result of a process of self organisation. The users of a language are willing to follow the grammar rules fully in order to communicate on the basis of clear, shared

understanding. Designers, on the other hand, want to affirm their own design language or at least allow it to evolve from some previously defined personal one. In natural languages, a grammar is an existing protocol shared by a community. In design, language is one of the products of the designer's work. It is an individual synthesis proposed to a community for discussion. This inversion explains why most uses of shape grammars relate to analytical research, that is, work in which the main goal is to infer the set of rules underlying the generation of a family of designs. The Palladian grammar (Stiny and Mitchell, 1978) and the Prairie House grammar (Koning and Eizenberg, 1981) fall into this category. However, it seems clear at this point that in terms of designing a flexible system, the production of a language of designs, i.e. the design of a grammar, is a possible approach to this goal. In other words, if a designer designs the rules of a system, for instance, a housing system or an urban system, rather than defining a single design they are, in fact, proposing a system of solutions corresponding to the solution space defined by the grammar.
Shape grammars also allow us to predefine the way shapes are generated. For instance, if we separate into different parallel grammars those which generate things (or themes) represented by lines, from things (or themes) represented by polygons, such as lines representing street centre lines and polygons representing plots, buildings or blocks, we can ensure that each separate grammar always generates correct representations of each theme. This structure approximates the representations generated to those of typical geographic information systems. Description grammars can also compute aspects related to design features other than shapes.
As such, the core argument in this thesis is that **a design system based on compound forms of grammars can be used to design urban systems and simultaneously bridge the gap between CAD and GIS environments by providing the means for improving the analytical capacities of the design system**. This integrated process is envisaged as allowing designers to have an enhanced perception of the quality of their proposals, due to the analytical data produced by the generation system.
Developing a design tool involves solving the problem of using predefined design languages, namely the shape recognition and semantic difficulties. As such, this thesis also involves three technical research questions derived from the hypothesis of using shape grammars to develop an urban design generation tool.

How can we always guarantee a correct shape match in the design tool? [6]

How can we guarantee that designs are built meaningfully and in a GIS compatible format? [7]

How can we implement a grammar-based design tool without imposing a pre-defined grammar on designers? [8]

In order to solve the problems embedded in the previous research questions, a hypothetical technical structure for the urban generation tool may be supported by several grammar formalisms and an ontology. The grammar formalisms involve the

use of parallel compound forms of shape and description grammars to generate design moves, i.e. design actions which urban designers perform recurrently while designing. Designs are obtained by combining these design moves. The ontology provides the semantic structure. An ontology, according to Gruber (1993), is a formalisation of a shared conceptualisation. In this case the conceptualisation addresses the urban domain. In formal ontologies a specific domain can be described by object classes containing objects or concepts from that domain, moving from generic to detailed concepts and including a specification of the relations between object classes. It is the hierarchy of relations between object classes that builds up the semantics of the knowledge domain in question. In this case, the object classes correspond to representations (shapes and descriptions) of components or elements of the urban space. The representations found in the object classes constitute the description and shape sets of the grammars. The semantic structure of the ontology is therefore transferred to the grammars, creating more meaningful behaviour in the generation process. To be more precise, the ontology defines the relations between city representations by providing an overall semantic structure for representing cities, whilst grammar formalisms such as labels (Stiny, 1980), weights (Stiny, 1992) and colours (Knight, 1993) may be used to control the more localised or contextualised details of the generation process. The details of this structure will be explained in Chapters 5 and 6 of this thesis.

§ 2.4 Hypothesis

The main driving force behind this research addresses the problem of planning and designing for the complex behaviour of cities.
Flexibility is proposed as the means of dealing with the problem of flexibility in the design process and the design of flexible systems.
The research concentrates on the development of tools for designing flexible and adaptable urban systems.
To this end, a hypothesis has been formulated:

A generative urban design tool can be defined using parallel compound forms of shape and description grammars that will be able to:

- **Generate alternative urban designs according to a set of predefined goals;**
- **Generate designs in formats which are compatible with a GIS environment;**
- **Generate descriptions of the designs in terms that complete the understanding of the design (e.g. functional descriptions, density indicators, hierarchical descriptions of streets or other urban elements).**

The flexibility of designs is therefore defined by the rule set used for the generation of alternative solutions instead of the typical single layout.

The technical structure of the urban design generation tool required to solve the latter technical issues encompasses the following structure:

1. The definition of parallel compound forms of shape and description grammars encoding algorithms in the form of reusable design patterns replicating typical design moves. Designs can be customised by combining design moves in different sequences and arrangements.
2. An ontology containing concepts describing the urban space and the urban design process, establishing the semantic structure of the design tool through their expressed relations. The design patterns capture the semantic structure of the ontology by using the shapes and descriptions representing the concepts found in the ontology.

If a complete solution is provided to the problems posed by the three questions stated above, this thesis will have contributed to shape grammar studies in the following ways:

- By contributing an additional formalism or a device to solve the matching problem for urban design purposes in GIS compatible formats;
- By contributing to the definition of shape grammar-based design tools which are able to generate semantically meaningful designs;
- By contributing to the development of grammar-based generation tools that allow for the synthesis of personal design languages without imposing predefinitions.

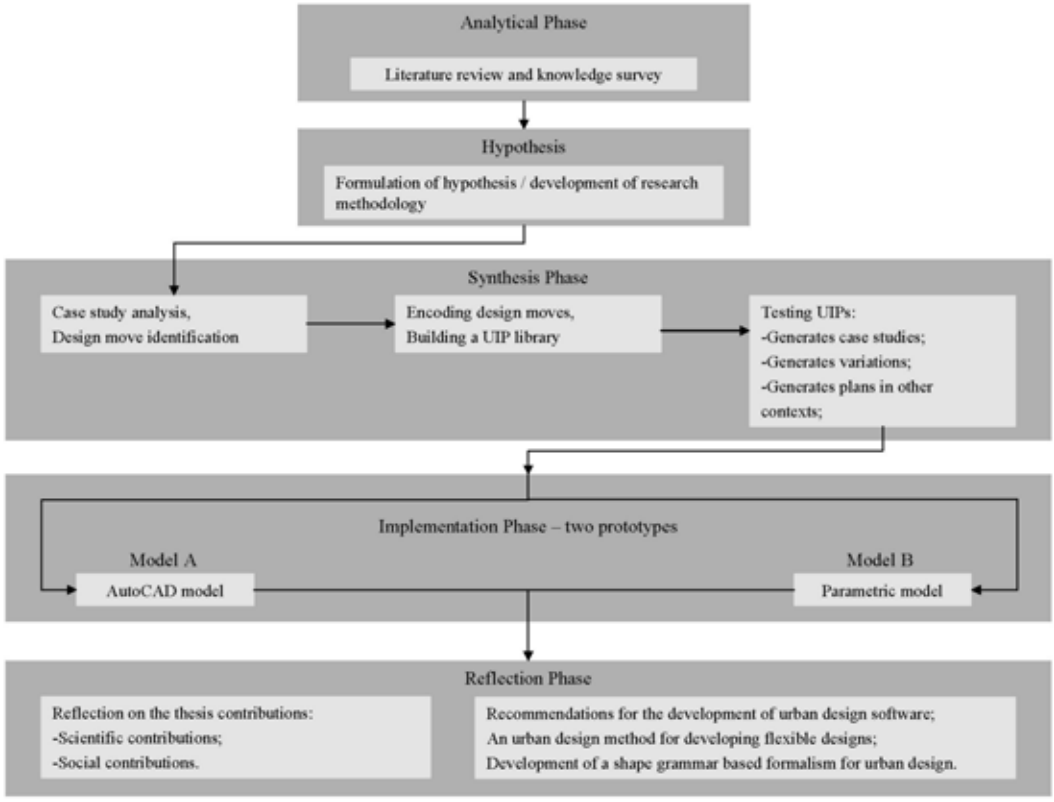
The theoretical model and its prototype implementations will be called CItyMaker and will constitute the generation module for the City Induction research project. CItyMaker contains the acronym CI for City Induction and the acronym CIM for City Information Modelling. The purpose of the latter will become apparent from reading the thesis.

3 Research methodology

The previous chapter presented the research problem addressed in this thesis. To summarise, the research aims to find appropriate tools and methods for designing efficient and flexible urban plans as a response to the unpredictability of city development. As previously explained, flexibility is cited by many authors as a strategy for addressing the complexity of city development. The main idea is to produce flexible designs developed through an interactive design process. Flexibility is envisaged on three levels: the capacity to accommodate changes in the programme of requirements (*design flexibility*); designing systems of solutions (*flexible designs*); the capacity of these systems to adapt during its life span (*flexibility of the design*).

This chapter explains the research methodology used to solve the research problems identified in the previous section. After the main problem had been identified, the work was divided into five phases: the analytical phase, involving a literature review; hypothesis formulation, in which the knowledge gathered was used to formulate a hypothetical solution; the synthesis phase, in which the theoretical model and design method was developed, based on an analysis of case studies; the implementation phase when two prototype implementations were developed as proof of concept of the theoretical model; and the reflection phase, in which critical reflection on the results of the conceptual model and prototypes provided a set of recommendations for the implementation of software tools for urban design.

The following diagram provides an overview of the research workflow showing the five phases defined above.



1 Analytical Phase – during the analytical phase an extensive literature review was carried out in order to gather enough supporting information on the research problem.
2 From this information the Hypothesis was formulated: an urban design tool using grammar-based design patterns replicating typical design moves could be used to generate flexible plan layouts.
3 The Synthesis Phase involved developing the theoretical model to support the definition of a design method and the structure of the supporting design tools. This phase followed a three step sequence:
   - Analysis of a set of case studies in order to identify the design moves executed by the designers (a).
   - Encoding the design moves into Urban Induction Patterns (UIPs), a grammar-based generative formalism for encoding urban design moves. The UIP formalism was defined as a way of building up designs in an iterative fashion, move by move, until the design is complete. UIPs compute city objects (shapes and descriptions) found in an ontology co-relating concepts describing cities. A formal definition of UIPs was synthesised as described in Section § 5.6 (b).

- Validation of the concepts defined in the previous step. This was accomplished by showing that the UIPs developed were able to generate the plans on which they were based, to generate variations on those designs and to generate new urban designs for other contexts (c).

4 In the Implementation Phase two alternative interpretations of the theoretical model were developed: Model A, a prototype developed in AutoCAD; and Model B, a parametric version developed in Rhinoceros and Grasshopper. The implementations were shown to the authors of the case studies for comments and critiques.
5 The Reflection Phase involved critical reflection on the research achievements, identifying:
   - Reflections on the pros and cons of the design methods and tools developed, identifying their main achievements and shortcomings;
   - Recommendations for urban design software development;
   - Reflections on the scientific and social contributions of the research;
   - Identification of future work.

§ 3.1 The analytical phase

The analytical phase involved a knowledge survey and literature review of the research problem. The survey focused on the following themes: urbanism and the complexity of cities, design methods, urban design, geographical information systems (GIS), CAD and GIS software, computer science, pattern languages, shape grammars and description grammars, and ontologies and knowledge bases. Previous experience involving the use of patterns and shape grammars as described in (Beirão, 2005), (Beirão and Duarte, 2005), (Beirão and Duarte, 2009) was also considered significant knowledge at the beginning of the research, as well as information regarding the case studies used in this research. The analytical phase also encompassed the identification and selection of supporting case studies, including a data survey of the case studies selected to identify design moves used by urban designers. For this purpose the authors of the plans were also interviewed, with the aim of understanding their concerns and design intentions and, above all, their design moves, i.e. their sequence of individual design decisions, in order to understand how the plans had been synthesised.

The analytical phase involved two main approaches:
1 investigating the knowledge areas and all the technical details related to the research problem and sub-components of that problem;

2 capturing the knowledge embedded in the design process of the authors of the four case studies.
3 The former provided an in-depth understanding of state of the art knowledge and the latter an insight into on the specificities of the design decisions taken by designers while designing their plans.

§ 3.2 Hypothesis formulation

The hypothesis was based on the information collected in the analytical phase. The hypothesis was built on the premise that all the research problems identified would be answered with the same solution, namely, that the hypothesis should respond: (1) to the complexity of urban design problems; (2) to the peculiarities of urban design, namely to an unpredictable phased and participatory process; (3) to the need to develop flexible urban systems, (4) to the technical constraints of the existing support systems, namely GIS environments for data analysis and CAD environments for design generation; (5) to an algorithmic structure that could be codified into generative codes; (6) to the reflective process in design; (7) to the need to interpret different types of data from an existing context, produce several types of data as part of design solutions and reinterpret the data generated within this context. Each of these problems was first identified with a research question for which a particular hypothetical solution was formulated. The partial solutions were then addressed as a whole to define one theoretical framework capable of answering the problems posed by the research.

§ 3.3 The synthesis phase

This phase encompassed three steps:

1 Further analysis of the case studies in a rigorous detail in order to capture the details of design moves and translate them into shape and description grammars. This analysis provided the base knowledge for the development of the conceptual model and theoretical definitions. It included the development of an ontology describing the concepts involved in the urban design process, the structure of the generation model, including urban induction pattern and urban grammar formal definitions, typical design workflows and a design method to approach flexible

designs. The formal definitions provided a uniform structure for the design generation tool.

2 Identification and development of formally defined urban induction patterns, using the case studies and information provided by the authors of the plans on their design methods. The patterns were captured from the case studies and encoded as discursive grammars. The goal was to define a library of urban induction patterns to compose an urban grammar large enough to support the generation of a wide variety of urban designs. The grammars were defined in ways that enable them to be reused for generic purposes rather than just for reproducing the design solutions that they originated from. When combined in various sequences by applying different parameters, they are able to generate a multitude of design solutions.

3 Demonstration of the application of urban induction patterns, showing that they are able to generate the case studies, parametric variations and alternative solutions for the case studies, and new urban plans in different contexts. The demonstration was achieved by showing the design rules of UIPs (i.e. their grammars), different derivations of those rules and, essentially, how they can be applied in the generation of different plans.

§ 3.4 The implementation phase

The main goal of the implementation phase was to develop the prototype computer implementations of the concepts developed in the previous phase. Two different implementations were developed, illustrating different approaches and exploring separate aspects of the same theory:

1 Model A, developed within the City Induction project framework, followed the previously defined conceptual structure rigorously. A software platform was chosen for this implementation, considering the best platform that could support the integration of the three modules in City Induction and the specific features needed by each module. At the end of each generation, the set of rules used defines the space solution and therefore, the flexibility space of the plan. This model is a rule-based model generating consistent representations and data which can easily be used in a geographic information system to analyse the solutions. The ontology-based representations and the parallel grammar structure guarantee GIS interoperability. Model A was developed with the help of grant holder Gelly Rodrigues at TU Lisbon.

2 Model B followed a parametric approach to a similar way of addressing urban design but focused on exploring the best possible interaction between design generation and design information flow, expressed in terms of outputs from

density indicators and derived indicators. In this model there is continuous interactivity between the model and the designer. The interface is very dynamic but the model is essentially parametric. UIPs are defined as design patterns (Woodbury, 2010). Model B was started with Pirouz Nourian at TU Delft. The design pattern structure was later improved with the help of grant holder Pedro Arrobas at TU Lisbon.

At the end of the implementation phase, the results were compared to identify the advantages and disadvantages of each approach.

§ 3.5 The reflection phase

The reflection phase provided the discussion and conclusion for the results of the research, focusing on its contribution to science and knowledge in the fields of urban design, design support tools, and shape grammars.
As a concluding statement, the reflection phase provided results on three different levels:

1. A set of recommendations for developing software for urban design, namely in terms of how it should be structured to support GIS interoperability.
2. A design method to enhance the quality of information flow supporting design decisions in an urban design process and ultimately improving the overall quality of design decisions. This includes reflections on how the method and tools may be used in participatory processes.
3. A tool to support studies on the relationship between urban morphology and density.

The reflection phase included thoughts on how the proposed approaches would contribute to the development of more sustainable cities.

4 State of the art

This chapter will address state of the art knowledge with regard to several themes involved in the research, providing an overview of all the subjects relating to the research topics, namely:

§ 4.1 Previous work: the use of shape grammars in urban design
describing previous experiences in the use of shape grammars in urban design studios;

§ 4.2 The complexity of the urban environment; the role of participation and its methods
providing an insight into how participation affects the urban design process and what urban design tools should do;

§ 4.3 Patterns, Pattern Languages and Design Patterns
surveying the concept of patterns, its use in design and how object oriented programming has adapted the concept to develop modular and reusable algorithms;

§ 4.4 Shape Grammars, Description Grammars, Discursive Grammars and Semantics
presenting these concepts and their generative properties and discussing the main achievements of grammars in design studies, focusing on their generative properties, namely their capacity to capture the design language underlying a particular set of rules;

§ 4.5 Design machines and the design process
explaining the structure of a design machine and comparing it with the design process in order to determine whether it is possible to integrate design machines into a creative design process;

§ 4.6 Ontologies and CAD-GIS interoperability
focussing on the role of ontologies in capturing the concepts underlying a particular domain and defining the relations between them; discussing how the specification of formal relationships between concepts can be used to promote efficient CAD-GIS interoperability;

§ 4.7 Measuring the urban space – some thoughts on formulation and evaluation
introducing objective methods for measuring density and calculating density-based indicators, and discussing how these indicators can be used to understand certain urban space properties.

§ 4.1 Previous work: the use of shape grammars in urban design

The use of shape grammars in design has expanded in recent years, either adapting the original teaching methods of Knight (1999) or adapting them more or less freely to new types of design methods and problems. Shape grammars can already be found in teaching programmes at ETH Zurich, in Switzerland; Unicamp, Campinas and UFRGS, Rio Grande do Sul, in Brazil; Carnagie Mellon University, Pittsburgh, Pennsylvania, MIT, Cambridge, Massachusetts, in the US; TU Lisbon, in Portugal; and Yildiz Technical University, Istanbul, in Turkey, to mention some of the best known examples. Regarding the use of grammars in urban design, a few research programmes should be mentioned:

- CityEngine is a grammar-based software for generating cityscapes that was developed by researchers from ETH Zurich and is being used in their urban research programmes (Halatsch, Kunze, and Schmitt, 2008), (Jacobi et al., 2009). CityEngine is not shape grammar-based but considers a procedural grammar (Parish and Muller, 2001); (Muller, 2006).
- At TU Lisbon a design studio programme has been running since 2001, involving the use of patterns and shape grammars. As this has been one of the main driving forces behind this present research, the following paragraphs will provide a detailed overview of the studio practices and the research based on the outcomes of these design studios (Beirão and Duarte, 2009) (Duarte and Beirão, 2011).

Regarding the latter point, the papers cited provide a detailed survey of the work developed between 2002 and 2004. The papers focus on how patterns and shape rules were used together and the advantages of using the inherent algorithmic structure of patterns in the design process (Beirão and Duarte, 2009). The design studios were divided into two semesters, the first addressing urban design and the second the development of customisable housing systems. In the first semester, students were introduced to a design method using shape grammars and pattern languages as an approach to the challenge of designing urban space using adaptive and flexible systems as a means of tackling the problem of unpredictability in urban dynamics. The first design studio dealt with the creation of a planning strategy for a large development area in Portugal, with strong development expectations resulting from the construction of a dam. It was expected that this area would experience radical transformations in the coming years due to the potential economic development resulting from the very large artificial lake created by the dam. In this context, students chose whether to expand the existing villages or build a new town in a strategically chosen place. They were organised in teams and asked to use a design method involving the use of patterns and shape grammars as instruments for designing a flexible plan for 5,000 inhabitants. The pattern language was suggested as a tool for

developing the urban program. The proposed design method had been inferred from an analysis of urban plans produced by acclaimed designers and an analysis of their design methods (see Beirão, 2005). This method required a four-phase approach to urban design in which the design rules would be defined in four levels of detail. These levels involved (1) identification of territorial features that could establish the plan guidelines and the rules for producing them, (2) the rules for designing the urban grid or grids, (3) the rules for designing urban units, considering these to be identifiable units of urban elements from the neighbourhood to the urban block and (4) the definition of rules for designing the plan details. Shape grammars were presented as a possible formalism for expressing the design rules at any moment in the design process. The main idea was to foster a more conscious perception of the relevance of rules in planning the urban environment, rather than enforcing any commitment to a design layout. In other words, rules were used to design the desired flexibility of the plans, therefore expressing flexibility on four levels of detail. These four levels are also consistent with the normal stages in urban planning and their respective approval procedures.

In the second design studio, in order to simulate real world conditions, in particular the role of phasing approval procedures, and to force students to consider urban planning and flexibility on different scales, the work was structured in four parts: acquisition of theoretical knowledge, urban analysis, urban plan design, and detailed plan design. In the first part, lectures were given on Alexander's pattern language (1977), Stiny's shape grammars (1980), and the four-phase design method. In the second part, they were asked to analyse the site, taking existing regulations into account. They were also asked to select a set of patterns from Alexander's pattern language to guide their way through the design towards specific programmatic goals. In the third part, they were asked to design an urban plan for a large expansion area in the northern sector of a town with approximately 25,000 inhabitants by developing shape rules to use in the generation of the design. In the fourth part, the detailed plan was developed, based on an urban plan defined by a different group of students in the previous phase. A copy of the assigned urban plan, together with its rule set, was given to each group which they then used to design a layout using the rules. The idea was to generate detailed alternative solutions by exploring the underlying flexibility within the limits set by the larger scale plan, thus emulating a real design situation.

The results proved that the rule-based design approach helped the students to deal with complexity and flexibility issues. Although the students did not use automated generation of the rules but rather basic 'copy and paste' manipulations of predefined blocks with some eventual additional routines, the approach had the advantage of avoiding the usual difficulties resulting from the technical implementation of shape grammars. Thus, although the generation process was slow, they could easily resolve gaps in the rule structure as well as transform the rules to explore design possibilities further. The process produced a high level of complexity in the proposed design solutions which would not have been possible in one semester without the aid of these design methods (see Figure 3). There was underlying flexibility in both the urban plan

and the detail plan, and it was also evident in the fact that most teams redesigned the urban plan following the design rules but proposing an alternative layout that would fit their detail plan intentions better.

Figure 3
Two examples of plans designed by students using a design method based on patterns and shape grammars.

The two design studios involved a total of 19 teams – 9 in the first design studio and 10 in the second – who produced the 19 urban plans that constituted the body of analysis supporting the research.
In the majority of the plans, students resorted to Alexander's Pattern Language. It became clear that one or two patterns played an important role in configuring and characterising the urban solution, both in morphological and social terms. Some patterns were recurrent and were used by almost every team, although only a few had a real recognisable impact on the proposal. The selection of a few patterns proved to create a strong sense of urban characterisation, giving the design proposal a clear identity. In addition, the patterns were open to designer interpretation. The students could later express the patterns through design rules explaining how to instantiate them. The recurrent patterns they used included: community of 7,000; identifiable neighbourhood; neighbourhood boundary; web of public transport; ring roads; network of learning; nine per cent parking; parallel roads; sacred sites; access to water; activity nodes; eccentric nucleus; promenade; shopping street. However, some students felt the need to define their own patterns, usually based on some recognisable feature found in the urban context that they could pinpoint as recurrent, traditional or simply local, that could provide some contextualised input into the design. In such situations, they provided an Alexander pattern-like structure to describe their new pattern before providing the design rules to develop the design. Patterns were used as programming statements before developing the design rules to generate them. Through the patterns, the teams were able to express their development vision for the area in very generic terms. This was proposed and applied in subsequent studios as a formal way of

defining a generic urban programme consistent with Friedman's concept of development vision (1997).
Some interesting conclusions were drawn from this research, which can be summarised in 4 points:

1. A rule-based approach to urban design produces urban plans with implicit and explicit flexibility. Explicit flexibility is defined by the set of rules expressed to define the plan's solution space. Implicit flexibility is not expressed formally but underlies the rules followed by the design team in designing the plan.
2. Shape grammars and patterns contain the algorithmic qualities needed to develop formal generative systems for exploring urban design solutions.
3. Any urban design defined by an urban grammar in a design context is just one potential acceptable solution defined by the grammar for that context. The feasibility of a design is only dependant on the contextual data that constrains the design rules.
4. Designing with shape grammars and patterns leads to intentionally ordered principles in a planned area, whilst allowing for contained but wide freedom of design throughout the design process. The degree of flexibility corresponds to the design space defined by the rules.

The results of these design studios were very promising regarding the use of shape grammars and patterns in urban design. The research clearly showed the advantages of using them in education to foster awareness in students of their own design rules, enhancing the understanding of a design language and how to use such knowledge in the development of flexible design proposals. However, their use in practice with real support tools involves additional difficulties, namely in relation to the use of shape grammars and the characteristics of urban design practice.

§ 4.2 The complexity of the urban environment; the role of participation and its methods

Uncertainty and complexity seem to be dominant paradigms in the growth of cities. The main problem is that, even when planned, the development of cities is difficult to predict. Designing cities involves the ability to deal with many simultaneous and complex development behaviours and the components involved in such developments, and predict desirable and reasonably controllable city developments. Predictability in terms of city development has been shown to be virtually impossible to achieve (see page 35), because in complex non-linear systems like cities, prediction

and design are players in the system, changing its internal dynamics (Portugali, 2000). In a way, the assumptions that allow for any kind of prediction of a city's behaviour are affected by the prediction itself affecting the real outcome, thereby potentially deviating from the prediction. In addition, the constantly changing city dynamics in contemporary society has led to the growing inefficiency of the traditional layout planning approach, which is incapable of dealing with the necessarily fast response demanded by such dynamics. Flexibility and adaptability have become imperative as ways of addressing urban design (Ascher, 2001). Correa (2000) speaks of malleability and incrementality, referring to ways of addressing city growth in developing countries. According to Friedman, (1997) plans should prescribe a clear development vision on a very general and broad scale, whilst remaining flexible in terms of the design of specific urban spaces. Ascher mentions that the new urbanism should be a flexible urbanism that is aesthetically open, reflexive, involves active participation and, formally speaking, **an urbanism of devices able to elaborate and negotiate solutions rather than producing specific plans**[4]. In the design studios referred to in the previous section, patterns were the devices used to define a development vision and establish the programmatic goals for site development. Shape rules were the devices used to express design flexibility.

Traditionally, urban plans are developed using methods that aim to produce a single layout representing a rigid, definite solution. The plans are centred on the definition of tight and interdependent urban parameters that tend to reduce design to a direct formalisation of such parameters. However, legislation does not constrain design flexibility or the way in which it represents flexibility[5]. In fact, it does not impose

4 Summarised from Ascher's conclusion, page 85, Spanish edition (2001). My underlining.

5 The Master's thesis (Beirão, 2005) contains an entire chapter on the analysis of the Portuguese urban legislation. The conclusion of the analysis is consistent with this statement. However, the municipalities which are the main institutions responsible for planning and managing territory usually follow strict procedures and a common practice and do not explore the potential for designing more flexible approaches to urban design at all. Additionally, certain strictly administrative procedures such as statistics for taxation purposes impose specific ways of representing certain planned data; for instance, a precise number of dwellings needs to be declared. Therefore, although the legislation embeds potential flexibility and design freedom, some administrative procedures impose ways of presenting information that clash with certain expressions of flexibility. In my design practice, I have also experienced this kind of constraint a few times.

Some research was carried out into Dutch legislation and procedures regarding urban planning and urban design approval. However, it was decided not to include research on legislation. There were two reasons for this decision. The first concerns the fact that the research goals do not address specific contexts, but generic use for wide applicability. As such, legislation is considered a system of constraints which should work with an autonomous interface allowing a tool user to customise it according to the local regulations. The second reason concerns the fact that in many countries either the legislation or common practice might not be suitable for flexible approaches to urban design. The conclusion of such a study could be that the legislation is not adequate for the desired practice and needs to be corrected or rewritten. The first reason simply points out that a correctly developed design tool should be able to adapt to local conditions and the second that an extensive analysis of legislation concerns another line of research which, although interesting and useful, can be carried out independently of the present research.

specific representational devices, nor does it imply any specific way of designing. The usual rigidity derives from an unconscious repetition of procedures, probably because this makes it easier to design and to communicate design intentions. It may be added that this practice is also still tied to the old modernist design practice in which a plan layout was the artistic expression of an imposed formal approach based on the modern principles of functionalism. It can be argued that the current practice of urban design still lacks methods and tools to support flexible urban design.

However, it should be stressed here that the Dutch already have a long tradition of trying to approach urban design in new ways and specifically applying innovative design strategies to foster diversity as a means of enriching the qualities of the urban space. Additionally, their work on housing customisation and diversity seems to propose strategies that link all levels of detail in neighbourhood development and involve the participation of all kinds of potential stakeholders. For quite some time these kinds of approaches have been the subject of research, starting with the work of Habraken (1972), (1976) and have definitely had an influence on Dutch architecture and urban planning since then, especially through the work of the SAR – Stichting Architecten Research (Bosma, Van Hoogstraten, and Vos, 2000). Practices involving or deriving from this knowledge in Dutch urban planning have been well-documented in several recent publications such as (de Maar, 1999), (Boeijenga, Mensink, and Grootens, 2008), and (Theunissen, 2009). Nevertheless, although successful in terms of achieving diversity, Habraken's ambition to incorporate high levels of participation was never quite successful and is still a subject of research. The influence of the Dutch approach can still be seen in the work of many architects and urban designers (Gausa, Hammond, and Hammond, 1998).

In a traditional top-down approach, municipalities control the planning process through their hierarchical power as representatives of the citizen interests. However, this representative system seems to have failed in terms of the development of the modern city, producing highly criticised results and leaving the citizen outside the decision-making process. Participation is considered by many authors to be the best and most democratic approach to planning urban environments, involving anyone interested in the development of their city or neighbourhood. According to Arnstein (1969), true democratic citizen participation should be based on citizen power. However, the decision should be based on qualitative information, meaning that all possible means should be used to make information available to citizens in an impartial but technically accurate way. This is where the main role of the designer lies – the designer is a technical interpreter and new advanced tools may be used to improve the quality of such interpretations.

There are basically four kinds of agents involved in an urban design process: the design team, which will simply be called the designer; the local authorities, responsible for urban planning and management; the developers and stakeholders, who have legitimate interests in certain parts of the territory and represent the investment power and, finally, the citizens who are the users of the planned city. The citizens are the ones living the city, using it, benefiting from its improvements and complaining about its

failures and problems. They should be the most interested parties and therefore the most involved agents. Aware of this fact, most Western countries have been incorporating participatory procedures into their urban plan approval systems, but the real feedback from the population involved is usually very limited and viewed as a disturbance by the designers and urban management authorities.

Friedman (1997) proposes a schematic sequence for the urban approval process, as shown in Figure 4. There are four stages in this process: the development vision and concept code; the neighbourhood code; neighbourhood design; and neighbourhood construction. The process provides more freedom through a large gain in flexibility, reducing the time consumed in the process and the gap between the definition of the final design and the completion of construction. Spaces for decision-making and control can be found between these levels of scale. The codes defined at city and neighbourhood level should be subject to democratic decision-making through participatory processes. As Friedman suggests, once approved the codes can be used for administrative appraisal. Nevertheless, the fourth stage may, and should, be further sub-divided at a lower level of detail in order to distribute the design interests further at building design level. Such strategies are actually common in European countries, as opposed to the Canadian environment to which Friedman's work refers. The case of Holland, in particular, presents a very varied set of strategies for developing a greater degree of diversity in terms of building. This can be seen clearly in, for example, the Borneo-Sporenburg plan in Amsterdam (de Maar, 1999) or the Ypenburg plan in Delft (Venema, 2000).

Five levels of decision-making can therefore be considered: 1) Development vision and concept code; 2) Neighbourhood code; 3) Neighbourhood design; 4) Building design; and 5) Housing customisation, as shown in Table 1.

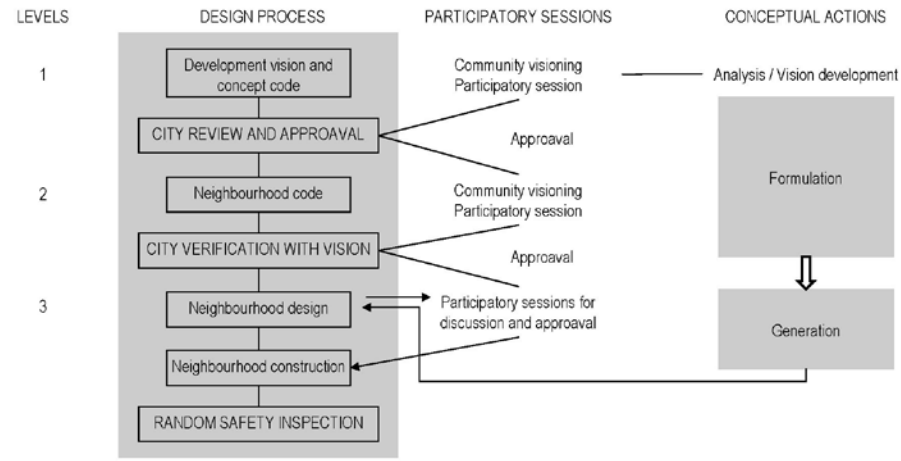


Figure 4
The role of municipalities in the approval process, as envisaged by Avi Friedman. (Source: (Friedman, 1997)).

Construction is not considered a decision-making level but simply a process of execution. The building work essentially concerns the architectural decision-making level but the building design level is the transition point, as it establishes the relationship between the proximity of private spaces and the city (public space) and therefore should at least be considered in the urban design process[6]. Ideally, citizens should be involved in all levels of decision-making with increasing intensity. Level Five should ideally be almost totally in the hands of the final user if all the common points have been taken care of in the previous stages. This thesis will focus on Level 2, with particular reference to neighbourhood design.

Level	Description	Scale
1	Development vision and concept code	Town / large district
2	Neighbourhood code	Neighbourhood
3	Neighbourhood design (neighbourhood construction)	Neighbourhood
4	Building design	Block to building
5	Housing customisation (building construction)	Dwelling

Table 1
Proposed levels of decision-making in urban design, based on Friedman's work (1997)

Urban design has certain specific features that differentiate it from other kinds of design:

- The process involves a number of decision-makers and is sub-divided into several decision-making steps which are not entirely dependent on the designer's decisions. This sub-division may occur for two different reasons: the scale of the design problem, and the particular requirements of participatory decision-making.
- The context has a considerable influence on the output format of the design, and is again dependent on the scale of the problem. Several output formats can be identified: the development vision (Friedman, 1997), a master plan, regulations or regulatory protocols, and detailed layouts, and all of which have particular variations depending on the context, local procedural regulations, the country's laws or local regulations. Most of these formats are expressed by communicating or restricting measures regarding the parameters of the urban space and urban indicators which, again, may involve particular contextual interpretations. As such, urban plans are expressed not just through layouts but also by urban indicators and measurements

6 The relationships between private and public space are a fundamental topic in several studies on urban space and are a recurring theme in space syntax research (Hillier and Hanson, 1984). The permeability of façades is a key element that affects the qualification of public space and should therefore be borne in mind by urban designers and other decision-makers.

used as restrictions or as design goals. In using these features, urban plans tend to express qualitative goals more than formal (morphological) goals.

- In architectural or industrial design clients may take the form of collective bodies or associations but they always involve a finite, well-defined number of actors, whereas in urban design they might be impossible to identify or predict in advance and might even change during the design process. As such, architectural design faces a well defined interlocutor whilst urban design has to deal with multiple and unpredictable interlocutors.

Participation provides opportunities for the democratic involvement of citizens as well as other stakeholders in decision-making during the urban design process. Amongst other advantages, participation allows for the integration of stakeholder concerns, a diversity of viewpoints, and greater acceptability of projects, mutual learning and mutual respect (Lach and Hixson, 1996). If well-managed, participation can improve results and save time and money. Community visioning is a common technique for community participation. Using proper supporting interfaces it must be implemented with the aim of achieving genuine levels of citizen participation – via partnership, delegated power or citizen control (Arnstein, 1969). Community visioning is achieved through open participation, which involves four steps called the visioning process (Steiner and Butler, 2007). These steps can be synthesised into four main questions: (1) where are we now? – defining the community profile from existing information and community values; (2) where are we going? – analysing probable trends and scenarios; (3) where do we want to be? – selecting the preferred scenarios for a community development vision; and (4) how do we get there? – plans, goals and strategies. The fourth question is the one that concerns the definition of the final design. However, the process of defining scenarios for questions (2) and (3) should be supported by at least a simulation procedure to help visualise the consequences of the decisions and should closely reflect real design procedures, focussing on exploring possibilities and therefore allowing for speculative, although grounded, approaches.
As stated in Chapter 2, Mayer et al (2009) propose the use of serious games in urban planning and simulation, stressing that this kind of tool should be integrative, dynamic, interactive, transparent, flexible, reusable and adaptable, fast and easy to use, communicative, educational and authoritative. However, it should be noted that there are substantial differences between simulating and designing. A design tool does not need all these qualities but will probably need a few others. Nevertheless, the advantage of combining the two processes as much as possible is that it approximates the final proposed design to the approved simulated scenarios. This has been the main reason behind serious research into gaming (Habraken and Gross, 1987), (Brandt, 2006), (Brandt and Messeter, 2004). The main difference between simulating and designing concerns the use of regulations supporting the urban plan. Regulations are simply a way of filtering out unwanted scenarios whilst designing. Simulation should, however, provide the possibility of assessing even absurd scenarios and as such, a simulation tool should be able to spot and check the limit conditions that determine

the shift between the acceptable and the unwanted. However, some authors say that working with physical objects engages participants more successfully in participatory activities (Moughtin, 2000), due to the size of the computer monitor or, if a projector is used, the distance created between the participants and the subject under discussion. To overcome this problem some researchers have developed games involving physical models as an approach to developing city simulation games for participatory meetings (Venhuizen, 2010) (Tan, 2009). Others have developed more immersive environments which allow for greater human-computer interaction (Kunze and Schmitt, 2010). Kunze and Schmitt present an unusual method for developing participatory vision. With the help of an interactive computer environment composed of large touch screens and tables, the participatory process involves two sequential procedures: context analysis to identify fact patterns (occurrences) and vision development to define concept patterns. The participatory process uses the interactive computer environment to enhance the analysis but most of the process is undertaken using pens and a card on which the participants record their patterns by writing and drawing simple schemas. The process is easy to run and can easily be understood by every participant.

Procedures for participatory methods are extensively tested and documented. Several methods already exist, have clearly defined procedures and are adequate for specific types of participatory events. Adequacy, preparation, means and methods are accurately detailed in the "Participatory Methods Toolkit: A Practitioner's Manual" which can be found online at [WS3].

§ 4.3 Patterns, Pattern Languages and Design Patterns

The concept of patterns has become quite successful since 'A Pattern Language' (Alexander et al., 1977) was published in 1977. In a complementary book (1979) Alexander further explains their uses and the ideas underlying the concept. Originally this concept proposed that typical problems occurring in the urban and architectural environment can be provided with a typical generic design solution, and that particular sets of patterns produce a design language for architecture and urban environments. The authors define a common structure for patterns in order to establish a standard format: an archetypal illustration; an introductory paragraph setting the context; a headline identifying the essence of the problem; a long section supporting the evidence for its validity; the solution description; the solution diagram; relations to other patterns. It has been pointed out that this structure facilitates a critical approach to content, allowing for refinement of the patterns and the creation of new patterns if the same structure is maintained. It provides an algorithmic structure for designing, due to

the fact that patterns in association with each other create a chain of interrelations as a result of their expressed relations to other patterns. The way the authors define the concept is abstract enough to be applicable in most circumstances and to be customised by the designers in order to adapt it to their personal design language.
For communication purposes, the conditional predicate → consequent statement can be represented in a schema identifying the problem elements, their attributes and parameters in a schematic representation of the predicate, and the transformations to these elements, attributes and parameters in a schematic representation of the consequent. Patterns, as defined by Alexander et al, are algorithmic structures which can be used to generate design solutions. However, the use of the word pattern has led to some misunderstandings regarding the structure of patterns. In current language, a pattern is a regular form or sequence discernible in the way in which something happens, is shaped or carried out, but it can also be considered a model that can be followed[7]. In the work of Alexander et al, patterns encompass both definitions: the former as the predicate and the latter as the consequent. It is the composition of predicate and consequent that gives the pattern its algorithmic structure. Recognising a recurrent occurrence in the environment simply involves finding a predicate. A recurrent design solution is also a pattern, in the sense that it constitutes a model to be followed. In this case the pattern is simply the consequent. Alexander et al's patterns imply establishing a relationship between predicate and consequent. The common misunderstandings involving the use of the term pattern stem from the fact that in Alexander et al the term pattern is used without any other adjective. This means that many people interpret it in a commonsensical way and use it simply for analytical pattern recognition, ignoring the algorithmic structure that gives rise to generative behaviour. Patterns, as defined in 'A Pattern Language', like shape grammars, can be used both for analytical and generative purposes.
Patterns have had many different uses in many knowledge fields, and computer science perhaps represents one of the most successful achievements. Aware of the algorithmic structure of a pattern language, Gamma et al. (1995), also known as the Gang of Four (GoF), proposed to develop this concept in a software design methodology called design patterns. This concept adds accuracy to patterns by adding a code sample for solving typical software design problems which enables the algorithmic structure to become rigorous and effective. They proposed a refined pattern structure divided into 13 sections (Vlissides, 2000): *Name* / *Intent* / *Also Known As* / *Motivation* / *Applicability* / *Structure* / *Participants* / *Collaborations* / *Consequences* / *Implementation* / *Sample Code* / *Known Uses* / *Related Patterns*. This design pattern structure can solve specific software design problems and make object-oriented programming more flexible and reusable. Their theory is cited as suitable for any object-oriented programming language. This upgrade to pattern language theory

7 Most dictionaries will corroborate this statement.

indicates a way of linking the mostly semantic aspect of the pattern concept developed by Alexander et al to a precise structure, suitable for computational purposes.
Several attempts have since been made to combine the formal approach presented in the Gamma et al design patterns with the Alexander et al patterns in order to develop computer aided design systems capable of generating design solutions based on the identification of a design problem (Salingaros, 2000) (Montenegro, 2010). However, architecture and urban design involve issues that are much too complex to be handled as linear tasks. In fact, these issues and problems are very much context dependent and many different formal solutions can be applied to solve a particular design problem. Therefore, the implication that something can actually be computationally generated implies assuming that there is a specific formal solution for a specific design problem, which was not entirely present in Alexander's idea. In fact, Alexander avoids indicating specific formal approaches in order to free design space for designers and this is a consistent theme throughout his publications. We should therefore distinguish between the very generic and essentially conceptual patterns present in Alexander et al's theory and the very specific problem solving patterns defined by the GoF. We are in fact facing two similar concepts with different foci. Alexander's concepts are generic, flexible, open to idiosyncratic interpretation, conceptual and defined on a high level of abstraction, whilst the GoF's patterns are specific, directed towards problem solving and defined formally as algorithms with a code template that enables solutions to be generated for a locally specified problem. Both approaches to the concept of patterns can be useful in terms of defining an urban design tool. They may provide the formalisms to control a generation system from a high level of abstraction to a high level of detail.

§ 4.4 Shape Grammars, Description Grammars, Discursive Grammars and Semantics

A shape grammar is a set of shape transformation rules that are applied recursively to generate a set of designs (Stiny, 1980). Formally speaking, shape grammars are algebras of the form $\{S, L, R, I\}$ where S is a finite set of shapes, L a finite set of labels, R a set of transformation rules and I the initial shape. The transformation rules have the form $\alpha \rightarrow \beta$ in which α and β are labelled shapes from the set of shapes S and the set of labels L. The rule finds the occurrence of a transformation τ of the labelled shape α in a design δ and replaces it with a transformation τ of the labelled shape β as defined in the equation $\delta' = [\delta - \tau(\alpha)] + \tau(\beta)$, where δ' is the resulting design after the rule iteration and − and + are the Boolean difference and union operations. To put it simply, a rule finds a subshape α in a design δ and replaces it with a new shape β. As Stiny has pointed out, subshape recognition is an ambiguous task (2005) and needs correctly

supported artificial intelligence to be effective in a computer-based implementation of shape grammars. Finding $\tau(\alpha)$ can prove a very complex task when new shapes emerge during design generation. In addition, extending shape grammars to include parametric grammars, which are in fact used in most design situations, makes this even more difficult, as the recognition of a shape becomes the recognition of any assignment *g* of parameter values to a parametric labelled shape α, i.e. finding $g(\alpha)$ in a design δ to apply the rule schemata $g(\alpha) \rightarrow g(\beta)$. Matching the shape $g(\alpha)$ in the left-hand side of a rule in a design δ is hard to accomplish. Although the formalism has quite an accurate mathematical foundation, shape recognition in these terms is still computationally a very hard task and the use of shape grammars has been limited to date, due to the difficulties of implementing shape grammar interpreters. However, shape grammars are a useful device for elucidating the composition of designs in terms of the spatial relations defined between their sub-shapes or sub-designs.
Description grammars (Stiny, 1981) provide descriptions of designs in terms other than shapes, referring to purpose, function, meaning, type or other. A description grammar is defined in terms of a description function $h : L_G \rightarrow D$, mapping designs in a language L_G defined by a grammar G to descriptions in a set D. The description function allows a shape to be associated with a description, for instance, its function, in such a way that it can provide a meaning for that shape. Stiny (1985) proposes using a combination of shape grammars and description functions for computing designs with form and meaning. The relationship between description and shape grammars is established by the use of compound or parallel grammars in which shape grammars produce recursive transformations of shapes generating more complex shapes, and description grammars produce recursive transformations of descriptions of designs which might correspond to changes in meaning, function, components or other elements in the descriptions set. The association of shapes and descriptions is usually established by the description rules. Excellent examples of the use of description grammars in parallel with shape grammars to control meaning in the generation of designs are given in (Duarte, 2001), (Li, 2001) and (Knight, 2003).
Shape grammars have successively demonstrated a capacity to encode the design rules embedded in design languages with a rigorous technical formalism (e.g.: (Koning and Eizenberg, 1981); (Stiny and Mitchell, 1978); (Buelinckx, 1993) and their use in the analysis of design languages has proved quite efficient and accurate. However, semantic discourse in urban design is not only provided by shape transformations but also by political, social and territorial contexts, which are informed by features other than those of form. Shape transformations are, in fact, a result of reactions to the latter features rather than simple relationships between shapes. The failure of shape grammars to deal with semantics has been previously pointed out by Fleisher (1992). Fleisher's argument is interesting and still valid in some aspects. All his criticisms concern issues of semantics. However, as previously mentioned, description grammars allow their users to deal with design languages in relation to features other than form. Duarte (2001) has addressed this kind of semantic issue, proposing the concept of discursive grammars, a combination of description grammars, shape grammars and

heuristics as a way of generating designs that are appropriate for a particular set of contextual features. The main concept is an adaptation of the design machine concept (Stiny and March, 1981). A discursive grammar is composed of a programming grammar and a designing grammar. The programming grammar finds the occurrence of particular features of a context and uses a set of description rules to generate the specifications of a design programme. These specifications are then used to trigger the application of the design grammar, which finds the correspondence between descriptions of solutions and the shape rules required to generate them. A set of heuristics is used to guide the generation into the solution space. Nevertheless, Fleisher's main argument concerns a kind of inversion of the origin of creativity which underlies the origin of shape grammars. A shape grammar presupposes the existence of a language of designs. Its use for analytical purposes is therefore quite consistent as it enables the rules underlying design styles or languages to be understood. Fleisher says that grammars in natural languages are the result of analysing how grammars organise syntactic rules from self-organised natural languages. In the case of design, the language should be the result of designing, at least if we consider designing a creative act, whereas in natural languages, the language and therefore the grammar are the a priori accepted agreement that allows for communication. Therefore the idea of using grammars for design synthesis contains a consistency problem because during synthesis the design language is not a known premise but just one of the products of the design process. It is actually the one which allows the design to be explored even after the conceptual design is complete. This is probably also the reason for the lack of success of shape grammars amongst design practitioners. However, it should be said that the results of the design studios referred to in Section § 4.1 (page 60), and the students' positive reaction to the use of shape grammars supports the belief that the concept could be used for designing and that some advantages can be derived from the generative features of shape grammars. What should be understood from Fleisher's criticism is that there are still methodological issues to be solved regarding the use of grammars in creative design.

Shape grammars, like description grammars and discursive grammars, are particular types of production systems (Gips and Stiny, 1980). The word 'grammar' as a mathematical construct was first used by Chomsky (1957) as a way of understanding and generating sentences in natural languages. The main misconception underlying the use of the term 'grammar' in the field of design concerns the notion that design is a problem-solving activity. Although consistent with many researchers (Newell and Simon 1972), from the time shape grammars were developed, this notion contradicts other viewpoints that focus much more on design practice, such as those of Donald Schön (1983) or Brian Lawson (2006) – see also Section § 4.5 , page 74. When confronted with design practice these models are much more consistent with common experience than the idea of problem solving. The problem with this misconception is that it is assumed that we can correctly describe the goals of a design by being able to correctly describe the problem that created the need for the design. However, as Lawson and Schön point out, designers often are not completely aware of the all the

components of the problem and their awareness is often based on trialling hypothetical solutions or individual design moves, through which they find substance for reflecting on the deeper hidden particulars of the design problem. It is this progressive structure that creates a progressive awareness of the design problem. If shape grammars have no way of dealing with this characteristic of the design process, they will be no use in supporting designers. Furthermore, one specific characteristic of creative design that is frequently cited is the notion that certain solutions emerge unexpectedly during some experimental move performed by designers in the course of exploratory procedures. How such emerging design qualities can be reorganised is, computationally speaking, an extremely difficult artificial intelligence problem to solve. However, as previously suggested, combining several kinds of production systems in parallel or compound rules can at least partially solve the semantic problems. The remaining semantic problems may be solved by allowing for human decision-making during the design process.

§ 4.5 Design machines and the design process

A design machine (Stiny and March, 1981) defines an algorithmic structure for design and it is composed of four parts: a *receptor*, an *effector*, a design language and a theory. The *receptor* establishes the relationship between the outside world or context and the system and is supposed to provide descriptions given by a finite sequence of symbols encoding information on the outside world. These descriptions are called design specifications or programmes. The *effector* produces an object or design according to a set of design specification descriptions. The design is generated using a design language which provides a set of candidate designs and a set of descriptions of the candidate designs. The theory establishes the relations through which our understanding of the context can be compared with the candidate designs and thus provides the link for fitting designs to specifications. It represents the value system in the design environment.
A design machine implies a clear understanding of context, or, more precisely, that we know in detail which data the *receptor* should extract from the outside world and how to interpret that information. Although valid for problem-solving, this is not always the case with design. In a design process the understanding of the design context (the outside world) and even the specification of a design problem evolves throughout the design process as a continuous upgrade fostered by continuous reflective actions of the designer, his own analysis and moves. Every move implies an appreciation of the state of the design before and after the move, and a quality judgement on this (Schön, 1987). Furthermore, design synthesis emerges from a progressive awareness of its rules, which means that a design language cannot be known at the beginning of the

design process, at least if we accept that some creative decisions need to be made. On the contrary, the design language is built up during the design process.
Donald Schön discusses the practice of architectural design as an arrangement of micro design decisions – moves, in his terms - following reflective see-move-see cycles (Schön and Wiggins, 1992). See-move-see cycles have certain implications: firstly, that a design is a sequence of reflective cycles, secondly, that each move is the consequence of some local evaluation and, thirdly, that each move is followed by a reflective action on the instantiation of that move, i.e. that there is some local evaluation of the instantiated design. Seeing implies an appreciation of the circumstances in a design for which the designer is able to recognise a mismatch with some design goal or contextual feature that enables him/her to react by applying a 'move experiment' to upgrade the quality of the evolving design. Therefore, an effective design system can only be created if it contains a reflective structure.
As such, a design machine would appear to contradict the fundamental concept of the reflective design process unless we focus on the concepts that are able to clearly reproduce defined design actions in which premises (descriptions of the problem) and goals (descriptions of solutions) can be correctly formulated. Similarly, isolated *see*-move-*see* cycles may have a structure compatible with that of a design machine if what we *see* in the beginning and what we *see* at the end can be predefined. This is equivalent to a design pattern in which a 'move' corresponds to a generative algorithm. The important thing to stress here is that these concepts are complementary.
Whatever use may be made of the concepts of design machines and design patterns in computer science, their use in creative design may only be perceived as useful for this purpose if a reflective structure is maintained in their application. However, in order to do so there is a need for a detailed understanding of how creative design works and how it is distinctive from problem solving.
Lawson (2006) proposes a very ingenious model for explaining the design process, stating that design is negotiation between problem and solution through analysis, synthesis and evaluation (Figure 5a). One particularly interesting idea in this concept is that analysis, synthesis and evaluation are not seen in any particular order, since feedback loops are likely to occur in any direction (Figure 5b). Another interesting aspect is the presence of a solution at any moment in the design process, incorporating the somewhat controversial concept of Jane Darke (1979) that a hypothetical solution, a primary generator, is put forward early in the design process as a beginning for a problem-solution negotiation process or simply as a way of restricting the design space to a manageable framework. It should be noted that this is consistent with Knight's (1983a), (1983b) idea of transformations in design languages. The hypothetical solution can be expressed in terms of an initial incomplete hypothetical language of designs which needs to evolve by developing rule transformations as the awareness of the design problem is progressively enhanced by confronting the changes occurring in the language. Lawson's definition seems to be very consistent with most situations involved in design practice but nevertheless focuses on architectural design, whereas

the concern of this thesis is urban design, which involves other aspects that impose particular workflows, methods and participants. Section § 2.2 explores this subject.

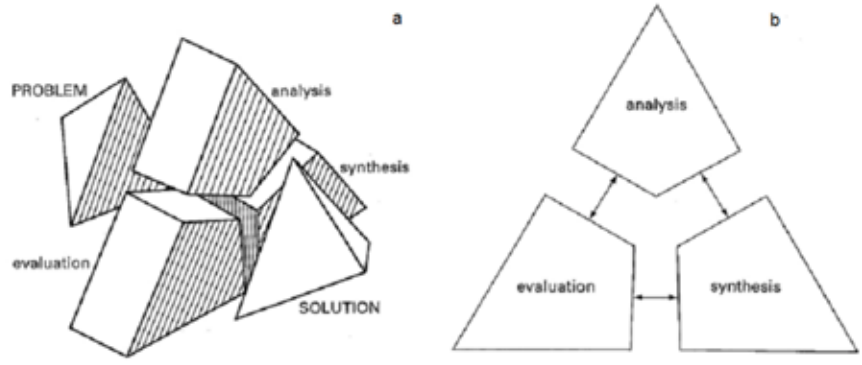


Figure 5
Design as negotiation between problem and solution through analysis, synthesis and evaluation (source: Lawson 2006).

In terms of the complete design process, a design system can be defined if the following principles are respected:

- designs are obtained by applying a series of see-move-see design cycles that cover the full range of local design problems in which a whole design problem can be decomposed;
- the application of a design move implies recognition of a particular set of features found in the design context;
- agreements on generic goal concepts are available and the relationships between the generic goal concepts and the design moves are known.

These principles follow a structure such as that of Alexander's pattern language (1977), in which, although the patterns are arranged from general and large-scale down to detail and small-scale, their order of application is still not based on this top-down structure but on the interpretation of context, allowing different scale patterns to be applied. It may be said that Alexander's patterns are capable of capturing generic goal concepts by establishing the relationships between generic descriptions of design problems and generic descriptions of design solutions for the design problems. We can also understand design moves as patterns if we regard the concept as a see-move-see cycle. The best and most interesting aspect of patterns relies on the relationships expressed between the various patterns stated at the end section of each. They basically state that a certain pattern works much better in combination with certain others and the related patterns shown in the book are from both lower and higher

levels. This is responsible both for an algorithmic structure and for building up a system of relations.
To summarise, the main goal of this sub-section is to present the set of interrelated but still disconnected forms of knowledge required to automate the generation of designs: design machines, which postulate an algorithmic structure for the computational production of designs (Stiny and March, 1981), a model for the design process embedding a negotiation process involving subjective assumptions (Lawson, 2006) and designs seen as arrangements of design decisions on two different levels of scale, conceptual decisions created by using patterns (Alexander et al., 1977), and locally evolving decisions made by using design moves (Schön, 1983). Finally, a structure similar to the design machine concept but occurring at micro level can be recognised in the design move cycle concept. In addition, shape grammars can be recognised as an available algorithmic formalism suitable for executing the transformations (synthesis) that constitute the design move.

§ 4.6 Ontologies and CAD-GIS interoperability

Geographic Information Systems (GIS) manage large amounts of geographical data in different formats – representational, either raster or vector, and alphanumeric data stored in several databases. All the data is linked together by means of a common geographical position. GIS platforms allow for complex analyses of territories by querying the databases to isolate particular aspects of this information and visualising them in several ways. In comparison with traditional maps, they work by accumulating all the maps and databases for a geographical area into a single compact platform in which all the data can be accessed and visualized separately using the interfaces provided by the GIS software. Common problems in GIS environments are related to the following issues: (1) data compatibility, (2) managing relations between different kinds of data, (3) maintaining links whilst performing design operations or, in other words, defining correct topologies with data representation, which is, to some extent, due to a further problem, (4) the lack of interoperability between CAD and GIS software. All four problems are related to the need for all the functions of both GIS and CAD software to operate using shared representations of the data being processed. Ontologies and knowledge bases are the most common computational devices used to deal with these problems. Both devices provide a way of organising data and retrieving knowledge related to the information stored. The term knowledge is used here, as the retrieved knowledge corresponding to the combined data obtained from relations specified in the database. The advantage of an ontology is that it enables relationships that have not been previously specified in the network of specified data to be inferred.

According to Gruber (1993), in computer science an ontology is a formal representation of shared concepts from a real or imagined domain and the specification of the relationships between them. Ontologies were developed as interoperability devices for sharing data. In the case of GIS ontologies they provide a protocol for a systematic classification of the concepts composing the geographic and urban environment. Ontologies aim for universal agreement within a particular knowledge domain (such as city or land use representation) and they are, by definition, always incomplete. From this point of view, an ontology is always an evolution of some knowledge domain and it is never finished but always available for further extension and refinement. However, the premises underlying the top level definitions of an ontology can sometimes be inadequate for modelling the structural concepts in the main concept domain of the ontology. If the top levels of an ontology do not describe the main components of a concept appropriately, the ontology will fail to perform its purpose and will have to be restructured from scratch because the main assumptions have not succeeded in structuring a correct description of the concept. However, replacing stems in an ontology with partial models allows new detail developments of the main concept to be defined. This means that, in general, ontologies are easy to develop and fine-tune in terms of detail levels but very difficult to work with if the core structure is found to be inadequate for specifying the concepts being modelled. Formally, ontologies establish a taxonomic hierarchy of the classes defining concepts in a domain, their attributes (properties or parameters) and the explicit relationships between them.

In a GIS environment an ontology is responsible for formally defining the shared concepts associated with the description of geographical information. CityGML (Kolbe, n.d.) is one of the most commonly used standards for the representation of cities. Although quite extensive in its description of the urban environment, there still appear to be some mismatches in CityGML with regard to other classification systems for city concepts, such as the street classification examples that appear in Marshall's "A First Theoretical Approach to Classification of Arterial Streets" (2002). Even though Marshall's document focuses only on the classification of streets and does not cover other city components it clearly shows, firstly, that it is difficult to be conclusive about this subject and also that although CityGML offers a very extensive classification of city concepts, it is still limited and not completely in tune with state of the art studies on the classification of streets.

§ 4.7 Measuring the urban space – some thoughts on formulation and evaluation

Understanding the urban space and the development of cities is in many ways related to understanding its measurements, the relationships between measurements and the relationships between measurements and corresponding morphologies. One common practice in urban planning is to explore the application of regulations based on different measurements of space. Density is probably one of the most commonly used measurements of urban space and in very general terms relates the amount of built space to the size (area) of a particular urban site. It is the ratio between the total amount of built area and the total site surface. In some cases density can also be defined as the number of people living and/or working in a particular area, but this definition is very ambiguous due to, for instance, variations in the average number of householders in a certain area as well as the size of the houses. Density measures give urban designers and analysts an approximate idea of the intensity of building work in an urban area. Conversely, the same measure can be used as a planning device to introduce specific building intensities into new proposed urban areas which are perceived of as beneficial for those particular contexts. This is because density is seen as providing a certain kind of qualitative information about an urban environment. However, as explained in Berghauser-Pont and Haupt (2010) density measures contain certain ambiguities and criticisms usually focus on two main aspects: (1) the scale of the urban area under consideration, and (2) the difficulty in pinpointing the relationship between density and urban morphology. The first criticism stresses that the scale of the area considered for analysis influences the density measurements. A large area measured might, for instance, contain green areas such as gardens and parks which will drastically reduce its density, whereas the density of even a low rise urban block might be very high in comparison. Another related point is that the definition of the boundary of an urban area is also ambiguous, as no conventions have been developed for this issue; if an area is bounded by a park, a street or a river, for instance, should it include their area, half of their area or simply consider the construction limits when making the calculations? Depending on the option chosen, the density measures will certainly be different. The second criticism is that a particular density value might have extremely different morphologies, as Martin (1972) has aptly demonstrated using the Fresnel diagram.

Berghauser-Pont and Haupt solved these ambiguities by introducing a few conventions for the measurement of urban spaces. To solve the scale problem they introduced the concept of levels of aggregation. These provide density calculations at building, plot, block (*island*), fabric and district level. The differences between the calculations are due to the difference in land base area between levels, which is called *tare space*. This tare space allows for a distinction to be made, for instance, between the total area of plots

in a block and the area of a block, which might also contain public spaces within its limits. Some conventions were also developed to solve certain ambiguities regarding the definition of a boundary of an urban area. They propose the use of three alternative methods to define boundaries using (a) administrative boundaries, (b) projected boundaries or (c) generated boundaries. Even though these conventions help to solve the majority of boundary definition problems a few ambiguities may still emerge in specific situations. However, once the boundaries are set, the density can be calculated objectively and the analysis based on the measurements can be carried out without involving other ambiguities.

Having set these calculation principles, Berghauser-Pont and Haupt developed some other urban indicators which together provide useful information for understanding urban morphology. They define a set of three basic indicators and a large set of derived ones. The basic indicators are: building intensity (*FSI*); coverage (*GSI*); and network density (*N*). *FSI* is a density value for a specific base land area and can therefore be calculated for any level of aggregation. *GSI* can also be calculated at any level of aggregation and refers to the amount of covered area in a site or, more precisely, the ratio between footprint and base land area. Network density calculates the ratio between the amount of construction area and the network length. One of the most interesting indicators derived is spaciousness (*OSR*), which calculates the amount of open space in terms of the amount of construction area. This provides some information on the intensity of usage of the open space.

All the indicators are calculated from four basic measurements: base land area (*A*); network length (*l*); gross floor area (*F*); and built up area or footprint (*B*).

Other indicators derived at fabric level provide information on parking performance or sunlight access for the fabric. The latter define average indicators providing information on the performance of the fabric and are only calculated at fabric level. However, spaciousness can be calculated at different levels of aggregation.

In order to rely on objective measurements and urban indicators, this thesis follows the measuring conventions found in (Berghauser-Pont and Haupt, 2010).

5 Defining an Urban Pattern Grammar

This thesis proposes a theoretical framework for developing urban design generation tools. The tool structure presented in the thesis is called CItyMaker and it was structured as an autonomous parametric and rule-based urban design tool. It is designed to establish a dynamic and interactive relationship with the user (designer). The tool was also designed as the generation module for the City Induction project, as mentioned in Chapter 1. This chapter defines the theoretical structure of CItyMaker: the ontology supporting the system and the definitions of the grammar formalism supporting the generative behaviour of the tool. Chapter 6 shows the methods involving their use and Chapter 7 shows two alternative computer implementations based on the same theoretical model.

§ 5.1 Summarising the hypothesis statement

The main driving force behind this research is the problem of planning and designing for the complex behaviour of cities.
Flexibility is proposed as the means of dealing with the problem, involving flexibility in the design process and the design of flexible systems (see Section § 2.2). Previous work involving grammar-based design and pattern-based design suggests they should be used to support flexibility. However, new tools are needed to improve interoperability between the analysis, synthesis, and evaluation activities during the design process, whilst simultaneously taking advantage of the generative potential of pattern and grammar-based design.
In order to maintain a regular design workflow, a design tool needs to maintain a high level of interactivity with the design team. Based on the observation that designers develop their designs in a progressive fashion, move by move, with each move a trial transformation of the existing conditions towards some improvement (Schön, 1987), this thesis proposes a design tool using a set of combinatorial algorithms – or design patterns – replicating the typical design moves that urban designers recurrently use. The proposed algorithms are defined in terms of parallel discursive grammars (Duarte, 2001) which are compound forms of shape and description grammars. The algorithms replicate design moves that are meaningful because they are part of a common and

shared design language used by most urban designers. During the research, these algorithms were given the name Urban Induction Patterns (UIPs). Designs are obtained by progressively combining UIPs until an end state is reached.

An ontology of the city and the urban design process was proposed to solve the technical semantic problems regarding the definition of the concepts used in urban design. An ontology is a formalisation of a shared conceptualisation (Gruber, 1993), in this case addressing the urban domain. In formal ontologies a specific domain can be described by object classes containing objects from that domain, moving from generic to detailed concepts including the specification of the relationships between object classes. The hierarchy of relationships between object classes builds up the semantics of the knowledge domain in question. In this case the object classes correspond to representations (shapes and descriptions) of components or elements of the urban space. The grammars use the representations found in object classes for their description or shape sets. The semantic structure of the ontology is therefore transferred to the grammars, creating more meaningful behaviour in terms of the generation process. To be more precise, the ontology defines the relationships between city representations, providing an overall semantic structure for representing cities, whilst grammar formalisms such as labels (Stiny, 1980), weights (Stiny, 1992) and colours (Knight, 1993) may be used to control more localised or contextualised details of the generation process.

This thesis outlines the structure of the ontology needed to define the urban design generation system and the set of generative design moves (UIPs) that provide the tools for designing urban plans.

The ontology is part of an ontology shared by the City Induction researchers. The shared ontology is being implemented within the City Induction group by Montenegro (2010) and it also contains the programme formulation model for the project. The ontology shown here was developed as part of the whole ontology and refers to the components found in the generation module. Parts of this ontology are shared with the formulation module.

The core of this chapter focuses on the definition of grammar-based patterns for urban design. The details of the structure will be described in the following chapter, whilst also exemplifying its use.

§ 5.2 Using patterns in urban design – a method and 3 design machines

The central concept of CItyMaker is the use of grammar-based patterns at different levels of abstraction. This section provides a brief overview of the proposed design system and how it is envisaged it should be applied in a regular participatory urban design process.

Alexander et al's (1977) pattern structure identifies descriptions of a design problem for which descriptions of a solution are proposed. If the descriptions of both problem and solution are formally translated into description grammars (Stiny, 1981), rigorous devices can be developed that are capable of reading formal descriptions of a context and generating descriptions of a solution in a similar formalism. The descriptions of solutions are obtained through a set of description rules. This can be achieved with the aid of a city ontology containing descriptions of the components of urban environments and specifications for the relationships between them. As previously stated, ontologies are suitable formalisms for this purpose. In this case, the ontology defines concepts describing the city, captured in object classes with particular attributes and parameters. The specification of relations between object classes defines the semantic relations between the components of cities. The object classes define the object sets (shapes or/and descriptions) which can be used by discursive grammars (Duarte, 2001) to generate urban programmes and urban designs.

CItyMaker was planned to be used as an autonomous design tool within the context of a design workflow starting from a participatory process and evolving to the generation of alternative design solutions. The structure of the design workflow is captured in Figure 6. It describes the structure of a pattern-based design model whose input is data defined as a set of patterns during participatory workshops – fact and concept patterns. The model is defined at neighbourhood design level. The proposed design workflow shown in Figure 6 presents a participatory process and a semi-automated process for programme and design generation.

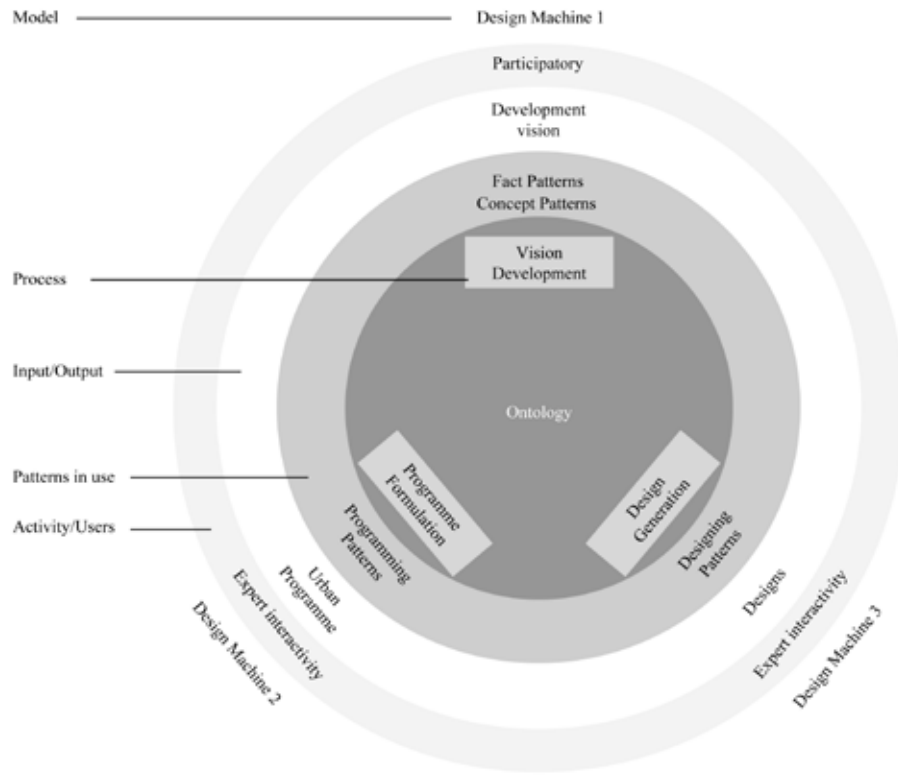


Figure 6

Pattern-based design model. The model is defined by three theoretical design machines. Design machine 1 corresponds to an entirely manual and interactive human procedure. Design machines 2 and 3 are semi-automated. A design interface manipulates both the ontology and the patterns.

Until the development vision is concluded, the design process is participatory, involving analytical and conceptual processes using fact and concept patterns. From programme formulation onwards, the processes are semi-automated. These semi-automated systems are production systems (Gips and Stiny, 1980), or design machines (Stiny and March, 1981) to be more precise, and therefore the patterns used in them have a precise algorithmic structure which can be considered as design patterns. The design patterns are distinguished by dividing them into programming patterns for developing the urban programme and designing patterns for designing the solutions (see Figure 6). The model is composed of three theoretical design machines which perform vision development, programme formulation and design generation respectively. Design machine 1 corresponds to an entirely manual and interactive human procedure. Design machines 2 and 3 are semi-automated.

A design machine is composed of a *receptor* and an *effector* (and a theory). The *receptor* is responsible for the analysis and the *effector* for the synthesis. The theory is half embedded in the ontology defined in the relationships expressed between object classes. The other half of the theory is expressed through the designer interactivity provided by the interfaces (ontology interface + design interface) allowing the characteristic ambiguity of the design process to be expressed.
Design machine 1 corresponds to a high level of abstraction in dealing with concepts which, although generic, are intelligible and usable by lay people. This process incorporates an analytical and a synthetic part. In the analytical part, the participants identify fact patterns or, in other words, context predicates. In the synthetic part, they build up concept patterns, or the desired consequents, which will be set as design goals. Both patterns are predicates that will be used in the programme formulation (Design machine 2). The rules that produce the concept patterns are implicit in the participatory process and do not need explicit translation. However, the predicates have to be inserted as input data through the ontology interface and this process is a matter for expert interpretation. The participatory process method is inspired by (Halatsch, Kunze, and Schmitt, 2009) and can be supported by several computational means, although decisions are basically made by the participants'[8].
The programme formulation machine contains programming patterns which are discursive grammars (description grammars + shape grammars + heuristics). The design generation machine contains design patterns which are also discursive grammars (description grammars + shape grammars + heuristics). The shape grammar part in the programme formulation part may be empty.
Both patterns and grammar rules have a similar structure (predicate → consequent). In the case of programme formulation, the predicate is composed of a development vision: occurrences = fact patterns + vision = concept patterns. The consequent is an urban design brief made up of descriptions of occurrences + descriptions of solution components. Note that in design generation (Design machine 3) the consequent of the programme formulation machine (Design machine 2) becomes the predicate of the design generation machine + an initial shape for activating the initial patterns. The consequents in the design generation machine are designs.
The level of abstraction is reduced from Design machine 1 to Design machine 3. Design machine 1 defines generic goals, Design machine 2 generates detailed descriptions of an urban programme, and Design machine 3 generates designs for the programme. The semi-automated processes benefit from designer interactivity, providing the characteristics for a reflective design practice as identified by Schön in his research on design methods and practice. Design machine 2 uses programming patterns defined by rules edited in the ontology and Design machine 3 uses designing

8 The possibility of semi-automating this process – Design machine 1 – is an option but this has been reserved as a possible evolution of the system.

patterns defined as Urban Induction Patterns (UIPs) (see the formal definition in Section § 5.6).
The following sections show how UIPs can be used to design urban plans and explore design flexibility.

§ 5.3 Case Studies

The urban design generation tool developed in this research bases its generative behaviour on the arrangement of sets of small design decisions, or trial moves, which progressively build up the whole design. It is understood that most designers use similar design moves to define their urban designs[9]. As such, in order to identify what these small common design moves are, some reverse engineering was required, using a set of urban plans as case studies. The design processes in these case studies were analysed, breaking them down into very small generic design moves. It was found that in the four case studies similar basic moves were used to generate the four different designs. After correctly understanding the structure of all the individual design moves, their parameters and design range, they were codified as urban induction patterns (UIPs) following the structure explained in Section § 5.6 .
Furthermore, the analysis showed that in order to produce complete urban designs at least four sets of urban induction patterns have to be defined, relating to four different levels of design: (A) rules to define the compositional guidelines of the plan; (B) rules to define grids or the main street structure; (C) rules to define urban units, including squares and other public spaces; and (D) rules for designing details, such as the detailed design of street profiles and materiality (Beirão and Duarte, 2009). These levels are also close to the regular stages for planning approval by local authorities. The levels come from the previous work described in Section § 4.1 .

9 Communication between designers illustrates this idea. Practitioners tend to develop a kind of personal slang which they use to communicate design concepts. Different levels of abstraction can even be identified in this slang. Some expressions are used only in a particular office and they have a meaning for the designers working there. Other expressions are common among designers sharing a particular design language. Moreover, all designers seem to be able to communicate with each other by referring to design moves which are common to the whole design community.

In order to define design moves in the form of UIPs, 4 urban plans were used as case studies (Figure 7). The 4 urban plans were: 1- The extension plan for Cidade da Praia in Cabo Verde by Chuva Gomes; 2 - Qta da Fonte da Prata, in Moita, Portugal by Chuva Gomes (QFP); 3 - IJburg/Haveneiland by Frits van Dongen, Felix Claus and Ton Schaap from a larger plan by Palmbout; and 4 - Ypenburg also by Palmbout (Palmboom and van den Bout). The case studies were used to frame the work within the scope of their design space.
The main goal was to use the case studies as a repertoire of examples from which a library of design moves could be inferred in the form of UIPs that could then be reused to design new plans.

Figure 7
Plans for (1) Praia, Cape Verde and (2) Moita, Portugal (both by Chuva Gomes), (3) IJburg, Netherlands (by van Dongen, Claus and Schaap based on a master plan by Palmbout) and (4) Ypenburg, Netherlands (by Palmbout).

The research started by analysing the first case study, inferring the design moves used by the urban designer and defining them as Urban Induction Patterns. In interviews he gave, Chuva Gomes clearly stated his design moves for both plans (case studies 1 and 2). He even pointed out which moves were common and which were different in the two plans. Basic definitions of UIPs were developed, based on his moves in such a way that they would be valid for the four case studies or even for other well-known paradigmatic urban plans. Due to its simplicity, Plan 1 provides very basic rules for

designing an orthogonal grid-based plan. The patterns were developed in such a way that in order to obtain results similar to those in Plan 2 the same rules (UIPs) could simply be applied with a different sequential arrangement and different values for the parameters. Roughly speaking, it can easily be seen that Plan 2 uses similar rules to those in Plan 1 but applies them several times in 4 different areas with 4 different orientations and parameter values (Figure 8). Introducing more complexity into the plans is therefore the result of combining basic UIPs in more complex ways. However, the two Dutch plans introduce new features suitable for encoding other UIPs. For example, Plan 3 introduces a lot of variety into the definition of the urban block and a set of additional rules could therefore be defined from this case study (see Figure 22– page 154). Case study 4 introduces an even more complex variety of design transformations in comparison to the previous, less complex, case study. In this plan greater variety can be seen in terms of urban unit definition, but distortions in the orthogonal grid also become evident, implying the definition of rules to deal with grid distortions or irregular grids.

The main approach was to start by inferring the design rules of the simpler case study and gradually address the more complex ones by exploring the parameters of the rules to their limits. The idea was to maintain the same UIPs as far as possible and simply relax the range of parameter application. In some cases, optional rules were introduced defining alternative solutions within a common algorithm. Options could be explained as alternative moves for a similar generic design instruction. In all cases, before developing a new UIP the range of use of the set of rules and their possible transformations within the pattern was tested in order to restrict the amount of patterns in the system to the minimum set required. This task proved hard, since it was possible to lose direction when considering valid variations simply due to the fact the design possibilities are infinite. To avoid this, the research remained focused within the scope of the case studies, aiming only to show that: (1) a minimum set of common UIPs could be used to generate the 4 case studies; (2) the same UIPs could produce new designs in other contexts. The generation of designs for other contexts uses the same set of UIPs. A specific language of designs can be composed from the set of available UIPs or, in other words, within the design space defined by the case studies. The additional capacities of UIPs outside this design space were considered only as additional qualities to support the concept, but were not established as a goal. The importance of this exploration could be verified by comparing the new design possibilities to patterns found in well-known acclaimed plans (for instance, Cerdá's Barcelona plan – see Table 14, page 196, showing the patterns used to generate Cerdá's plan), providing indications of how parameter manipulation of UIPs could be used to explore new design domains. Furthermore, if simple variations in UIPs easily suggested other possible design moves observed in well-known standard urban design procedures, these would be incorporated into the system as a way of extending the design space within a commonly accepted design practice. However, examples were always sought out to support such situations.

The information on the case studies was gathered from publications (Boeijenga, Mensink, and Grootens, 2008), (Mozas and Per, 2002), (Venema, 2000), online material (Projectbureau IJburg, 1996), (Den Haag, 2010), and elements and interviews that were kindly provided by the designers. The contribution of the case studies to the development of CItyMaker will become clear as the urban induction patterns are explained in the following sections of this thesis. Formal descriptions of UIPs including grammar rules can be found in Appendix 2 – A Library of Urban Induction Patterns.

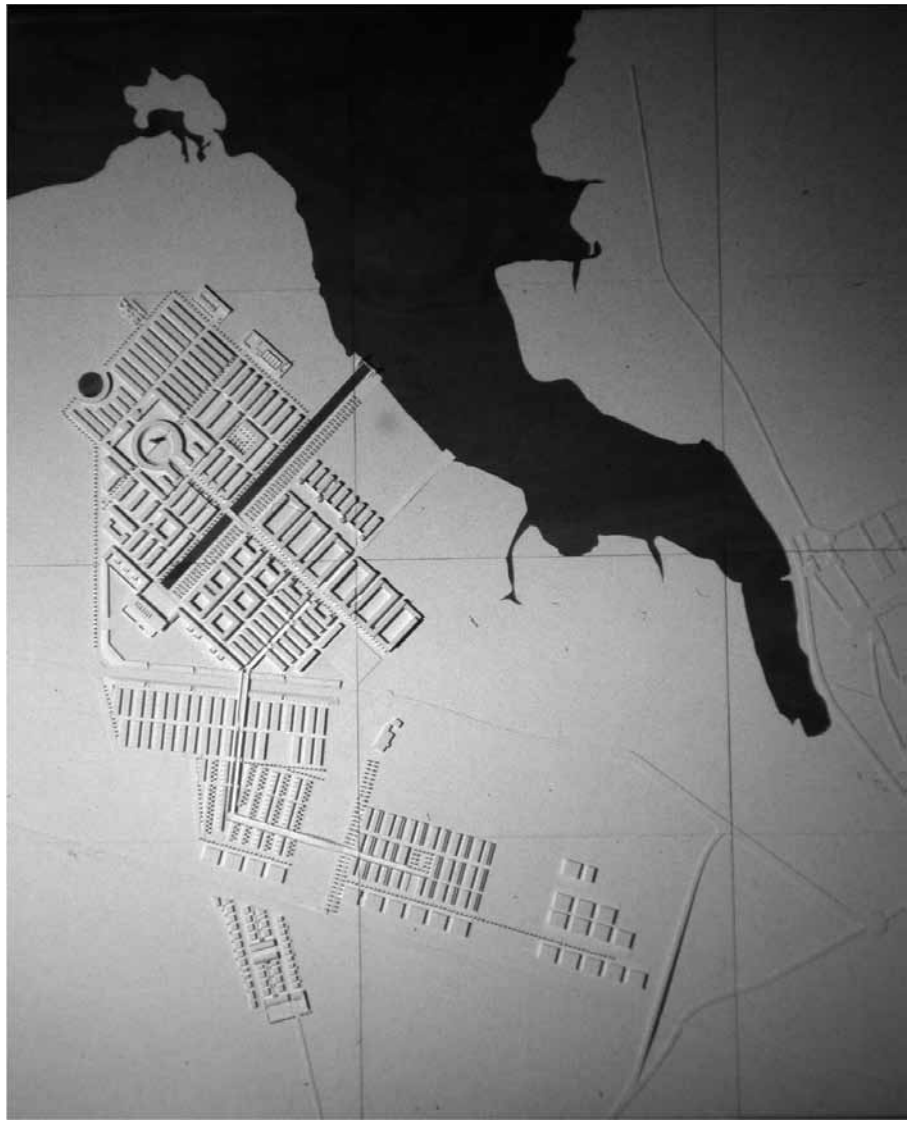


Figure 8
Case study 2 – A sequence of 2 orthogonal axes and a rectangular grid is applied in several areas of the plan.

§ 5.4 An ontology for the urban design process – the detail of the street system

Ontologies are formal representations of shared concepts from real or imagined domains and they include specifications of the relationships between the concepts. In the City Induction project, an ontology provides the definitions and relationships between the shared concepts needed to describe the city domain and the urban environment. This ontology defines and organises meaningful relationships between the various types of objects and features found in urban space and used in the urban design process. The city ontology is divided into sub-ontologies or systems, each one containing features from a specific domain of the city structure, namely 'Networks', 'Blocks', 'Zones', 'Landscapes' and 'Focal Points'. These 5 main classes define the top level of the ontology. It follows standard representations in geographical information systems that work with specific geometrical representations. 'Networks' are represented by lines, 'Blocks' and 'Zones' by polygons and 'Focal Points' by points. 'Landscapes' represent natural features as opposed to the artificial components of the environment represented in the other classes. As such, the three types of representations appear in 'Landscapes', separated into sub-domains. This top level structure is consistent with most 3D virtual city representation standards [WS9] and can be found with similar approaches in software such as the types mentioned in Section § 7.1 , CityCAD, CityZoom and CityEngine.

Regarding the vertical structure of a system or sub-ontology, 'Networks', for instance, describes the domain of connectivity and city morphology in which the street system may be identified. Systems in this context are autonomous semantic units within the ontology describing a well known sub-domain of the city. The street system is one such unit within the 'Networks' sub-ontology. Other systems can be considered in 'Networks', such as 'train networks', 'subway networks' or 'bicycle networks'. Some sub-domains may share some of their parts with other domains, for instance, the bicycle network shares most of its parts with the street network but they are definitely not coincident. Figure 9 shows the main structure of the city ontology, together with the basic classes and their relationships.

Systems are subdivided into object classes: each class has object types and each object has a set of parameters and attributes. The object types are defined through their shape representation and shape description, and they are instances of their respective object classes. Classes are denoted by two bold capitals. The systems are part of the ontology, as branches or interlaced branches of it, depending on the specific relationships defined between classes. Each system has a particular meaning in terms of the understanding of cities. For instance, cities may be referred to as a complex network of streets, a street system, or a property system. Each system has a particular kind of representation associated with the kind of information being depicted.

The representations in each class can be geometrically defined by points, lines or polygons individually. Figure 9 also indicates the primary and secondary relationships between object classes. These two kinds of relationship establish two levels of priority which can be used for semantic manipulation of design rules.
As an example, the next section details the street system. The same principles may be used to define and detail the other sub-ontologies.

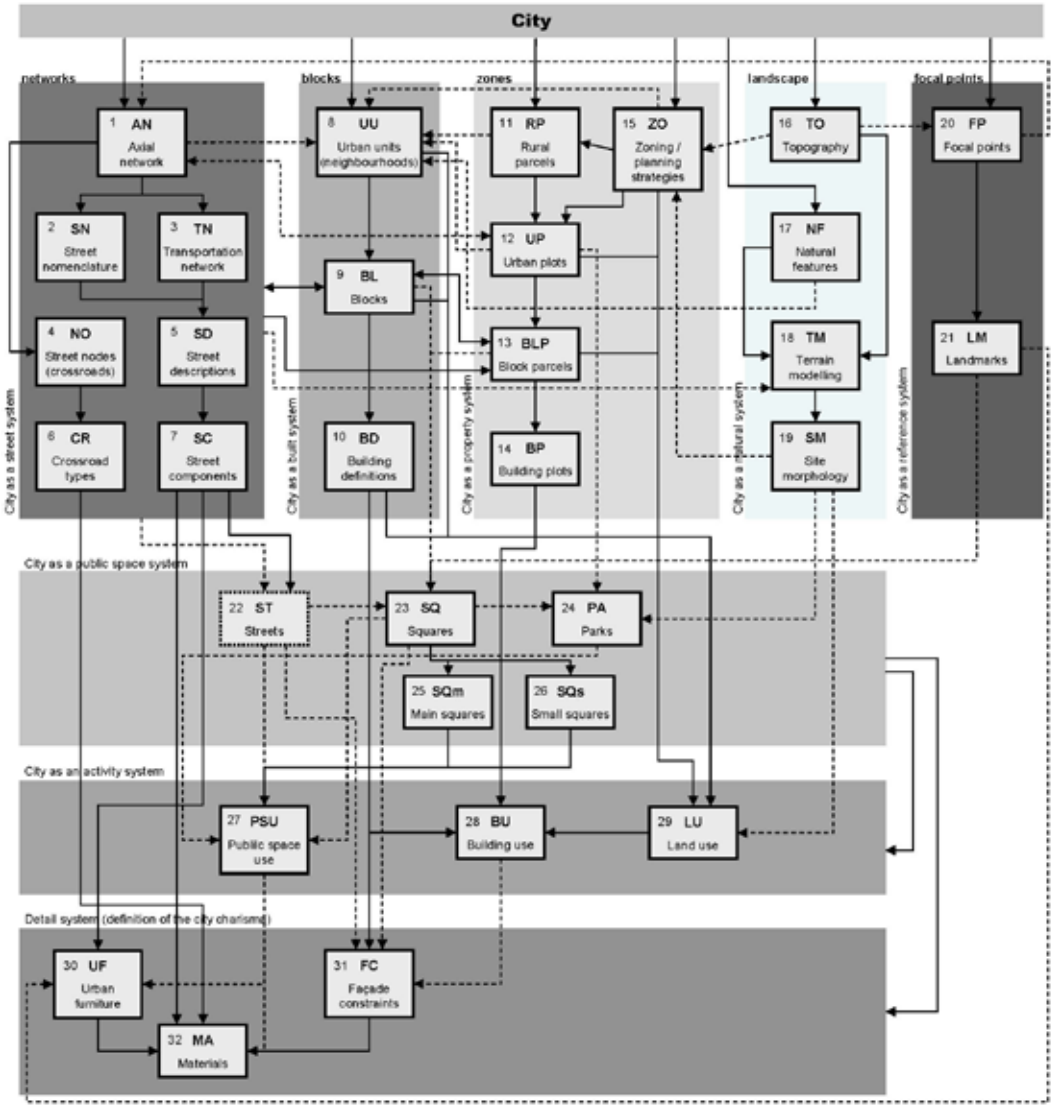


Figure 9
The main structure of the city ontology identifying the main systems and top level classes. Dashed lines indicate secondary relationships.

§ 5.4.1 An ontology for networks

'Networks' are a first level branch of the ontology used in the City Induction project. The street system is a semantic unit within the 'Networks' sub-ontology. The street system will now be used as a way of explaining the structure of the city ontology and its role in the generation model.

Street configurations may vary according to many factors, ranging from cultural and social to topographic or functional. Some may not be found at all outside a particular cultural context. Many researchers have tried to define and classify such characteristics but it would appear difficult to achieve a universal consensus. Marshall (2002) developed an extensive survey of the different ways of classifying streets and identified several different themes that have been used in street classification (Figure 10).

CItyMaker needed a simpler approach to the basic structure, an acceptable ontology for the street system that could be used for urban design to integrate concepts involved in the design process. In City Induction, the three modules imply different working platforms: formulation and evaluation are better executed in GIS whilst design performs better on CAD platforms. The underlying question concerns which descriptions and components of the street system should form part of a design tool integrating programme formulation, urban design generation, and urban evaluation. From the generation point of view, the ontology should additionally consider that representations of streets evolve in terms of detail and semantics throughout the design process, meaning that the ontology should contain classes that represent higher level abstract representations of streets before detailing them. The representation of streets should evolve from symbolic (axial, street centre lines with different hierarchies) to detailed representations by the gradual addition of meaning to the street classification until a detailed description of street components can be provided. Furthermore, the assessment of a street network in the early design stages should be an ongoing concern, as the street layout is probably the most permanent realisation of human artifice. As such, including different levels of abstraction in street representation should be a main feature of a design system and, in order to assess the network performance, all levels of detail should be amenable to use in a GIS environment.

The street system is divided into 5 major object classes involving a class of axial representations called Axial Network[10] (**AN**) which are compositional representations, a Transportation Network (**TN**) for the hierarchical and functional system definition, Street Nomenclature (**SN**) for a common language cognitive classification of streets, Street Descriptions (**SD**), providing a set of descriptions of the components composing the street types and Street Components (**SC**), a finite set of profile components for designing streets. Table 2 shows the permitted relationships between object types found in classes **AN**, **TN** and **SN**.

Index of Classification Themes

Part A Systematically Applied Classification Themes	1. Traffic Speed 2. Trip Length 3. Destination Status 4. Strategic Role 5. Circulation v Access 6. Administration
Part B Partially Developed Classification Themes	7. Network role 8. Access control 9. Traffic Volume 10. Transport Mode 11. Other Urban Users 12. Environment 13. Built Frontage 14. Road Width
Part C Diverse Themes	15. Street Name 16. Street in Cross-Section 17. Frontage Form 18. Planting 19. Street Character 20. Urban Character 21. Spatial Shape or Character 22. Visual Axis 23. Civic Role 24. Spatial 'Integration' 25. Urban Morphology (Formation) 26. Structural Role 27. Corridor Role 28. District Role 29. Land Use or Frontage Function 30. 'Towncentredness' 33. Urban Uses and Users 34. Living Space 35. Neighbourliness 36. Pedestrian Use of Streets 37. 'Diverse' Vehicular Classification 38. Public Transport 39. Sustainability

Figure 10
Index of classification themes for Street Classification (source: Marshall, 2002).

10 This should not be confused with the space syntax axial map. The axial network is defined by street centre lines and not axial lines, as defined by Hillier and Hanson (1984). However, the axial network creates a complete street topology and may be used for topological analysis using GIS tools, for instance. Whatever the tools used for analytical purposes, this is one of the main features that provides integration between the generation (synthesis) and evaluation activities.

§ 5.4.2 Axial Network (AN) – a hierarchy of compositional axes

The axial network is a symbolic representation of the street structure or a representation of compositional directions. Although we know that every street has a particular width defined by the buildings bounding it, it is common to represent them as lines on large scales. On a territorial scale, the width becomes null compared to the length. These lines represent networks of connections and also, from the designer's point of view, the composition lines defining the main guidelines and grids used to structure the design. The objects – lines – in **AN** are defined as compositional axes. a_1 to a_4 is a hierarchy of compositional street axes used to define the street network (see Table 2). a_b, a_p, and a_{tr} and a_{bu} belong to a thematically independent domain partially overlapping the street network. The bicycle network, for instance, can be defined as a continuous independent system possibly using reserved parts of the traffic system. They can be represented as autonomous networks in separate layers.
The objects belonging to **AN** can be further detailed in the design process, acquiring new meanings and roles in terms of defining the character of a street and the role it plays in characterising urban space. Depending on its role in the network composition, a street is further characterised by adding attributes which relate to its role in terms of a transport network or simply as an expression of city culture, defined in a word that identifies a particular type of street and its inherent street life.

§ 5.4.3 Transportation Network (TN) – functional representation for streets

It is common to find a hierarchical classification of street types defined in terms of traffic speed and other functional requirements. Marshall offers an extensive comparative study of this subject (2002), (2005). Despite the many different thematic approaches, it seems reasonable to say that when the classification is made exclusively from the point of view of traffic functionality it usually involves similar principles, although different names are used.
Pedro (2002a) defines the objective design requirements for public spaces in housing developments on a neighbourhood scale, including street hierarchy, detailing the quality requirements for streets on this scale and defining their relationships within the overall street system and public transport system. He defines four types of streets: main streets, distribution streets, local distribution streets and local access streets. The definition of the design parameters in this street hierarchy is based on the restrictions resulting from the maximum speed limit attributed to each type. These are functional criteria. The main interest in his approach is that he accurately proposes and justifies specific parameters

for every type of street and always specifies them for three levels of quality. The three levels of quality can be very useful in terms of introducing evaluation criteria for the whole network and may even allow for a dynamic evaluation of design quality throughout the design process[11]. These criteria can also be cross-referenced with some network indicators developed by Berghauser-Pont and Haupt (2010) such as network density and parking performance index, which relate street width to network performance at fabric level. Parking performance, for instance, is dependent on street width. A comparative study of street width, quality and performance could be developed combining this information for a better assessment of network quality and performance.

Axial Network (SN)	TN Classification	SN Classification
Composition structure		
a_1	**R1**, **R2**, **S1**, **S2**	**av**, **bv**, **ms**, **pr**, **gr**, **rr**
a_2	**R2**, **S1**, **S2**	**st**, **av**, **bv**, **ms**, **pr**, **gr**
a_3	**R2**, **S1**, **S2**	**st**, **av**, **la**
a_4	**S1**, **S2**, **S3**, **B1**	**st**, **la**, **al**, **cu**
Interlaced (with traffic) networks		**Can be part of...**
a_b Bicycle network	**B1**	**st**, **la**, **al**, **av**, **bv**, **ms**, **pr**, **gr**, **rr**
a_p Pedestrian network	**P1**	**st**, **la**, **al**, **av**, **bv**, **ms**, **pr**, **gr**
a_{bu} Bus network	**B2**	**st**, **av**, **bv**, **ms**, **pr**, **gr**
a_{tr} Tram network	**Tr**	**st**, **av**, **bv**, **ms**, **pr**, **gr**

Table 2
*Relationships between **AN**, **TN** and **SN** classes*

R2, **S1**, **S2** and **S3** in Table 2, Table 3 and Table 4 correspond to Pedro's classification: structural street, distribution, local distribution and local access, respectively. On a larger urban scale, the ontology needed to consider another street type above these offering long distance connections within the city, which we termed ring roads, following Alexander's pattern, **rr** in the **SN** object class and **R1** roads in the **TN** object class. Street types above this will not be considered here, although it is possible to discuss higher types for metropolitan interconnectivity. The equivalence between Pedro's classification and Marshall's stratification by speed (2005) is shown in Table 3.

11 In any case, even if the quality of the parts does not guarantee the quality of the whole, having qualitative criteria for street types and their parameters can at least enhance the designer's perception of the quality of the whole. However, the network qualities also depend on the topological relationships between the parts, which cannot be assessed in this way.

Transportation Network (TN)	Minimum requirements as a collection of profile components Street Descriptions (SD)	Allowed additional profile components (variations)	Stratification by speed (after Marshall)	Connectivity route types according to structural role (after Marshall)
Street types – transportation network				
R1 - High speed (ring roads)	⑦|⑫|**4x** ⑤*|⑫|⑦ * with central protection rail or green stripe	[③* ④** ⑤ ⑧ ⑩ b or s ⑪***] * with protection from car lanes, either ⑥ or ⑫ ** with no stops *** if protected from car lanes	S5 / S4	Corridor Cantilever Collector
R2 - Main Street / Structural Street	②|⑥|**2x** ⑤|⑥|②	[② ③* ④ ⑤ ⑥ ⑧ ⑨ ⑩ b or s ⑪*] * with protection from car lanes - ⑥ or ② if used for access to tram stops	S3.5	Connector Spine Collector
S1 – Distribution	②|⑥|**2x** ⑤|⑥|②	[② ③* ④ ⑤ ⑥ ⑧ ⑨ ⑩s ⑪**]	S3	Connector Spine Collector Cantilever
S2 – Local Distribution	②|**2x** ⑤|②	[① ③ ④ ⑤ ⑥ ⑧ ⑨ ⑩s ⑪**]	S2.5	Cantilever Cross-connector Stem
S3 – Local Access	②|⑤|②	[① ③ ⑤ ⑥ ⑧ ⑩s]	S2	Stem Cantilever

Table 3
***TN** object class – a hierarchical classification of streets according to traffic flow and functional characteristics (contains a comparison with Marshall's stratification by speed and connectivity route types).*

The ordinary citizen's perception of streets is essentially made up of symbolic features found in streets or the continuity of street spaces. Dimension, continuity, symbols and use, including relationships to buildings and traffic use, classify the different types of streets. Consciously or unconsciously these qualities are embedded in the common nomenclature attributed to streets.

Street nomenclature uses words from current language to express the semantic and cognitive classification, that is, names that embed some cultural information about the street life of particular kinds of streets. The main idea is that many of the words used to name street types contain a lot of information on their spatial characteristics and role in city life. Nevertheless, different languages use slightly different names or concepts for streets, as they can be extremely context dependent. In this research, the English vocabulary was used to select terms that could apply to all the case studies involved in the three City Induction studies and were common to most European urban structures. Particular street types could be added later, customising the ontology to fit particular contexts that had not been defined before.

The following words were considered representative of the different street types: street (**st**), avenue (**av**), boulevard (**bv**), promenade (**pr**), grove (**gr**), main street (**ms**), lane (**la**), alley (**al**) and cul-de-sac (**cu**) or impasse (Table 4)[12].
"street" is the most generic and abstract of these types, and is therefore considered separate from the others. "avenue", "boulevard", "promenade" and "grove" are defined as large thoroughfares with one or more lanes of trees or shrubs. The French terms "boulevard" and "promenade" used in several European countries tend to be associated with the largest streets. "promenade" is also defined as a leisure walkway usually including a large green area and leisure facilities. "grove" is usually associated with a greater densification of trees. A "main street" is essentially characterised by its social and commercial activity and therefore may have different configurations, often not planned at all due to informal development. The previous 6 types usually end in important public spaces such as main squares or junctions which may contain landmarks, buildings or other urban features such as monuments, statues or fountains. They correspond in most cases to higher hierarchies[13]. "Lanes" and "alleys" are small streets. "Lanes" might be associated with old rural paths embedded in the city street structure a long time ago. "Alleys" are more restricted streets, sometimes with dead ends, leading to interior neighbourhood spaces. The "cul-de-sac" or "impasse" concepts are dead ends with (or without) turning space. "Ring-roads" are usually perceived by people in more or less the same way as they are defined using the

12 Several other terms can be found in English to refer to street types, which are usually more specific and culturally grounded than these. However, because the system rules were based on Portuguese and Dutch case studies the more generic and internationally accepted terms seemed more feasible for the purpose of developing an urban design system that could be used on an international level. Marshall (2002) provides an extensive list of common English words used in street toponymy. Among the typical English terms, names such as crescent, mews or manor may be found, which provide information on the origins or shape of the street type.

13 Note that hierarchies a_1 to a_4 as defined for the axial network (Table 2) correspond to compositional hierarchies and do not necessarily have a direct correspondence to traffic use. The classification in the **AN** class attributes hierarchy to axes in terms of their compositional role in the design of the street network. The classification in the **TN** class is defined according to Pedro's (2002b) classification plus the high speed roads. Traffic use is a matter for analysis.

functional criteria, i.e. as large distribution streets for high speed traffic connecting different areas of the town. No matter how complex the classification of streets may be, it is important to establish a consensual classification that embeds the perception of streets in a common language that captures a certain social perspective on street classification (Marshall, 2002). In addition it must be generic enough to adapt to most contexts.
This classification can be used to provide some guidance for rule application in detailing the streets or (more interestingly) to provide information on functional distribution and the distribution of large public spaces in relation to other activities, mainly facilities and services.

§ 5.4.5 Street Descriptions (SD) – describing the street composition

The descriptions of streets found in the classes above are usually sufficient to provide an idea of the street layout, but not a detailed description. The generation module needs accurate descriptions to be able to generate detailed representations of different streets, even if their character can be generically described by the same word. Some interpretations of the same concept might have different representations within a valid range of parameters, which should reflect the designer's freedom of choice. By breaking streets down into a finite set of profile components, each street can be defined as a different arrangement of these components. The street descriptions are defined in terms of the minimum arrangements of their components. The minimum arrangement considered both in terms of quality levels and available street width will determine its final profile. Table 4 shows the minimum requirements for **SN** street types defined in terms of their minimum profile components and their possible correspondence as a **TN** object type when considered part of the traffic system or transportation network. The street components are shown in Table 5.

Street Nomenclature (SN) – a collection of street concept patterns	**Minimum requirements as a collection of profile components** **Street Descriptions (SD)**	**Possible relations to transportation network (TN)**	**Stratification by speed (after Marshall)**	**Connectivity route types according to structural role (after Marshall)**
st – street	②|⑤|②	**R2**; **S1**; **S2**; **S3**	S3.5 – S2	Collector Connector Spine Cantilever
av – avenue	②|⑧|2x ⑤|⑧|②	**R2**; **S1**; **(S2 + S1 + S2)**; **(S3 + S1 + S3)**; **(S2 + R2 + S2)**	S3.5, S3 With horizontal stratification S3.5 - S2.5 or S3 - S2	
bv – boulevard	②|⑤|②*|2x ⑤|②*|⑤|② * with tree alignment or green stripe	**(S2 + S1 + S2)**; **(S3 + S1 + S3)**; **(S2 + R2 + S2)**	idem	
ms – main street	②|2x ⑤|②	**R2**; **S1**; **S2**	S3.5 - S2.5	
pr – promenade	**(S2 +⑪+ S2);** **(S3 +⑪+ S3)**	**(S2 +⑪+ S2)**; **(S3 +⑪+ S3)**	S2.5, S2 With horizontal stratification	
gr – grove	**(S2** or **S3 + ⑪ + S2** or **S3)**; **(S2** or **S3 + ⑪)**	**(S2** or **S3 + ⑪ + S2** or **S3)**; **(S2** or **S3 + ⑪)**	idem	
la – lane	②|⑤|②	**S2**; **S3**; **P1**; **B1**	S2.5 – S1	Stem Cross-connector Cantilever
al – alley	② ; ②|⑤|②	**S3**; **P1**; **B1**	S2 – S1	
cs – cul-de-sac or impasse	②|⑤|②	**S3**; **P1**; **B1**	S2 – S1	Stem
rr – ring roads	⑦|⑫|4x ⑤*|⑫|⑦ * with central protection rail or green stripe	**R1**	S5 – S4	Corridor

Table 4

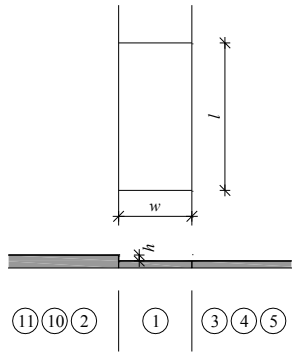
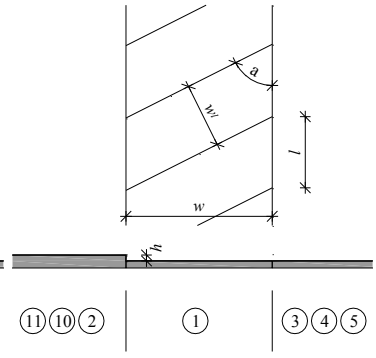
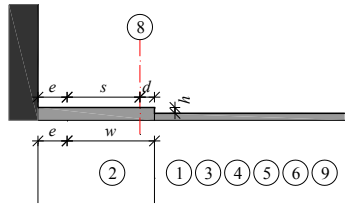
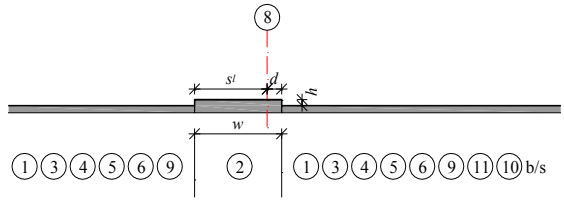
***SN** object class – classification of streets in common language. Comparison between **TN** classification and Marshall's stratification by speed and connectivity route types.*

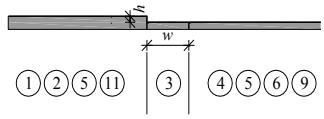
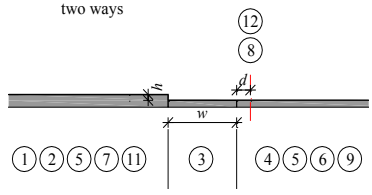
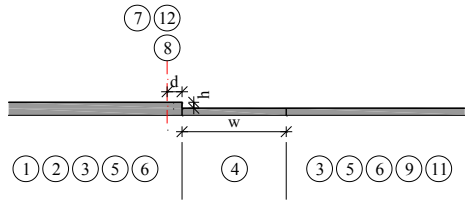
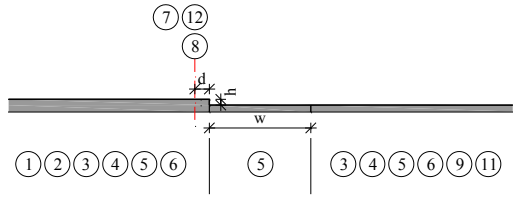
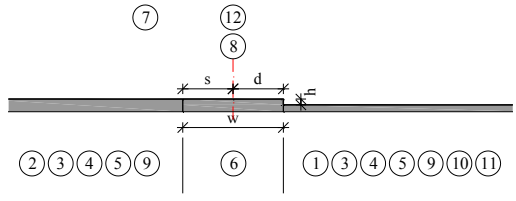
§ 5.4.6 Street Components (SC) – a collection of street profile components

The selection of components used in this object class is taken from the available technical literature, namely Pedro (2002a) and others (Marshall, 2002); (Steiner and Butler, 2007); (Burden et al., 2002); (Neufert, 2006), applied to the case studies used in the research in order to guarantee applicability. Curiously enough, the elemental components of streets are consistent in most of the technical literature and urban planning regulations, despite some insignificant variations in terminology. It is also relevant to point that the amount of minimal components is quite limited and common to most cultures, in spite of the fact that a component was needed to accommodate the Dutch canals that feature in the case studies. Table 5 shows the street profile components and how they can be used to define street sections. The parameters involved in the definition of each component are indicated, together with their range of variation and the relations that can be established with other components. The range of variation was taken from Pedro, considering his three quality level criterion to define the flexibility space for each parameter. Designers may, therefore, choose a target quality level by further constraining the flexibility space or explore different dimensions, receiving qualitative feedback in return. The values in a practical design system should be editable by the user, although the default values should provide a wide range of good standard solutions without the need to justify options outside the commonly used standards. The default level follows the intermediate quality level in Pedro's classification.

Finally, it is important to stress that the ontology shown here was developed to encode urban features within a design system or, in other words, was meant to encode urban structures for designing rather than describing the existing urban environment, and this may sometimes be inconsistent with the range of variation embedded in the qualitative definitions. However, the existing entities are accepted as existing representations with their own specific parameters and relationships, whereas designs, i.e. new representations, are constrained by the embedded pre-defined qualitative definitions, although they are still open to a wide range of parameter options and are, in general, editable. It is the role of the designer, through formulation and evaluation, to find the values that meet contextual needs, minimising the effect of pre-existing maladjustments. The main idea is not to over constrain the system and provide freedom to explore designs.

Typical parameters of specific urban types can be captured using data mining. This information can also be inserted into the ontology, adapting it to local conditions (Gil et al., 2009).

Street Components (SC) – a collection of street profiles	Profile schema – indicates profile parameters and possible adjacent profiles	Profile parameters
① - street parking Valid for S3 – S2 Preferred for S2.5 – S2	Parallel  Not parallel  	Parallel $2.0 \le w \le 2.4$ $4.7 \le l \le 5.5$ Not Parallel $4.8 \le w \le 5.5$ $2.25 \le l \le 2.5$ $45° \le \alpha \le 90°$ $0 \le h \le 0.2$ If ⑩ $h = 0$
② - sidewalks S1	 Central sidewalks  	$1.25 \le w \le 5.0$ $s \ge 1.2$ $w = s + ⑧width + d$ $0.3 \le d \le 0.75$ e is an extra space for additional purposes. E.g. – esplanade, benches, telephone booth, commercial activities, etc. $0 \le e \le 2.5$ And can be used also as tolerance The w value is further restricted depending on the street type to which it belongs. The values indicated here encompass all possiblities in the case of central sidewalks. For central sidewalks – $1.25 \le w \le 2.0$ $w = s^{l} + ⑧width + d$ $0 \le h \le 0.2$ $h = 0$ in the case of ⑥, ⑩ or ⑪

③ - bicycle lanes S1.5	one way  two ways  	One way $1.0 \le w \le 1.5$ Two ways $2.5 \le w \le 3.0$ Independent (not next to ④, ⑤ or ⑨) $2.0 \le w \le 3.0$ ⑫ in the case of **R1** apply for only with ⑥
④ - bus lanes S3.5 – S2.5	 	$w = 4.0$ ⑦ and ⑫ apply only to **R1** $0 \le h \le 0.2$ $h = 0$ when next to ① or ⑤
⑤ - car lanes Depending on relative position in cross-section S5 – S2	 	$2.5 \le w \le 3.75$ Variations depending on street type ⑦ and ⑫ apply only to **R1** $0 \le h \le 0.2$ $h = 0$ when next to ①, ④ or ⑤
⑥ - green stripes Can be placed next to any speed type	 	$0.8 \le w \le 4.0$ ⑦ and ⑫ apply only to **R1** $0 \le h \le 0.2$ $h = 0$ when next to ③, ⑩ or ⑪

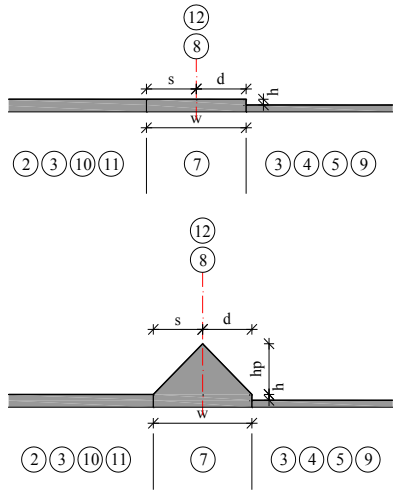
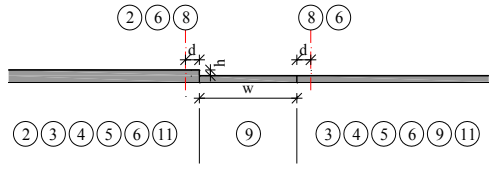
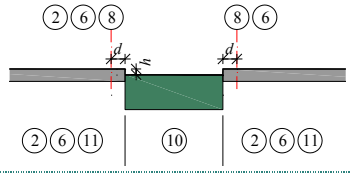
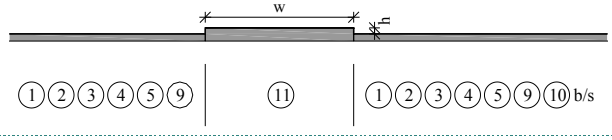
⑦ - noise protection (only for **R1** street type) To be used only next to S5 – S3.5		$2.0 \le w \le 10.0$ $1.0 \le h^{p} \le 2.0$ $Slope \le 45°$ The use of ⑧ or ⑫ in this profile is compulsory
⑧ - tree alignments	Tree alignments can be placed in ②; ⑥; ⑪ (according to parameters)	
⑨ - tram lanes S4 – S3		$2.5 \le w \le 4.0$ *w* varies depending on tram system. Width is fixed for each tram system.
⑩ - canal (**) big – b / small – s		*h* is variable depending on dam system $1.0 \le d \le 3.0$
⑪ - leisure walkway Can be related to any other except S5 and S4		$4.0 \le w \le 40.0$ $0 \le h \le 0.2$ For ② $h = 0$
⑫ - protection rails To be used in S5, S4	Protection rails can be placed in ②; ⑥; ⑪ (according to parameters) and only on ring roads	

Table 5
SC object class – table with the profile components of streets

§ 5.4.7 Implementation of the ontology

The ontology supporting the interoperability protocol between the modules in the City Induction project is also a main component of the programme formulation module. A prototype of the ontology was implemented by Montenegro et al. (2011), the researcher in charge of developing the formulation model in the project. The ontology was developed using Protegé, a computational ontology editor (Noy et al., 2001) which allows ontologies developed for a specific model or shared by the Web community to be imported, as well as for several ontologies to be merged from different sources into a more complex ontology or even for new ontologies to be edited from scratch. The ontology developed by Montenegro (2010) builds up an integrated and relational database of all the concepts involved in definitions of urban spaces. If additional rules are added to the ontology establishing regulations for mandatory dependencies between components of the urban space, it is then possible to obtain programmatic specifications for a particular context. The definition of this system, developed by Montenegro et al. (Forthcoming), constitutes the programme formulation module of City Induction, as well as the communication protocol between the three project modules.

§ 5.5 Patterns

The concept of patterns developed by Alexander et al (1977) in the book 'A Pattern Language' is particularly useful in terms of communicating design intentions. The key idea behind the concept is that typical problems occurring in the urban and architectural environment can be solved with a generic design solution, and that particular sets of patterns produce a language for designing architecture and urban environments. The concept provides an algorithmic structure for design based on associations of patterns that create a chain of interrelated decisions which can be applied in particular contexts. The way in which the authors define patterns is abstract enough to be applied in most circumstances and even to be customised to adapt to a designer's personal language.

Patterns seem quite reasonable and efficient urban design and planning concepts that can be used for communicating with both experts and lay people. Their structure, as proposed in the book, can be summarised as:

- a pattern name with an illustration of the pattern (an icon),
- a description of the design problem and its context (predicate),
- a description of the elements defining the design solution and their relationships (consequent),
- a set of interrelated design problems, i.e. a set of interrelated patterns.

Patterns are algorithmic structures that can be used to generate design solutions. The conditional predicate → consequent statement can be represented by a schema identifying the problem elements, their attributes and parameters (defined in a schematic representation of the predicate), and the transformations that will occur in these elements, attributes and parameters, (defined in a schematic representation of the consequent). In fact this is an identical structure to that of a shape rule in a shape grammar. The composition of predicate and consequent gives a pattern its algorithmic structure. However, the use of the word pattern has led to some misunderstandings regarding the structure of patterns. In current language, a pattern is a regular form or sequence discernible in the way in which something happens, is shaped or carried out, but it can also be considered a model that can be followed. Alexander's patterns encompass both definitions, the former as the predicate and the latter as the consequent. Recognising a recurrent occurrence in the environment involves finding a predicate. A recurrent design solution is also a pattern, in the sense that it constitutes a model to be followed. In this case the pattern is the consequent. These characteristics of patterns, as defined in 'A Pattern Language', can be used both for analytical and generative purposes just as with shape grammars.
Aware of the algorithmic structure of a pattern language, Gamma et al. (1995) proposed to develop this concept in a software design method called design patterns. This concept adds accuracy to patterns by adding a code sample for solving typical software design problems and making the algorithmic structure rigorous and effective and, more importantly, modular and reusable. Several attempts have been made since to combine both approaches in order to develop systems for generating urban design solutions on the basis of an identification of the design problem (Salingaros, 2000) (Montenegro, 2010). However, architecture and urban design involve issues that are much too complex to be handled as linear tasks, ranging from problem identification to solution generation. These issues are very much context-dependent and many different formal solutions can be applied to solve a particular design problem. Therefore patterns are reasonably abstract in order to keep a wide scope of application. In fact, Alexander et al avoid indicating specific formal approaches in order to free design space for designers.

However, at a micro level similar and typical design moves seem to recur in the work of designers, as can be seen in the four case studies used in this research. This characteristic is clearly present as a common language found in the discourse of urban designers. They all refer to starting with a definition of the main composition axis or main guidelines, then defining the grids (the orthogonal grid being the common denominator) and progressing towards the definition of urban units, such as neighbourhood requirements. Neighbourhoods contain a set of minimum components, such as the definition of public spaces, the definition of block types defining morphological regulations for specific block types and, finally, the definition of details for street profiles and other detailed aspects of public space such as materiality. This is absolutely consistent with the results of previous research developed by the author of this thesis focussing on other well known urban plans such as West 8's Borneo-Sporenburg plan, Siza's Malagueira plan (see (Beirão, 2005) and Manuel da Maia's plan for the reconstruction of Lisbon in the 18th century (Beirão, 2002). The latter example stresses the consistency of these generic methods over time. These consistencies in the discourse of designers may be termed design patterns for urban design, due to their recurrent use. Designers apply them in similar ways but each designer interprets their variables in a different way. Moreover, because every designer understands what each design move means in terms of the design process, an algorithm can be defined for them. Such an algorithm would encode the rules for generating the common design moves found in the work of designers and the designs would be the result of combining these algorithms into a whole in a semi-constrained way. This is the main concept underlying the structure of the design system proposed in this thesis. Furthermore, the rules of the design moves can be defined in terms of discursive grammars. The main formal definitions of such a system are discussed in the next sub-section.

§ 5.6 Urban Grammars and Urban Induction Patterns – definitions

The city ontology shown in Figure 9 is responsible for supporting communication between the concepts involved in the generation of designs. It also plays a fundamental role in communicating between the three City Induction modules: 4CityPlan (the formulation module), CItyMaker (the generation module) and EvModule (the evaluation module).
The top level of the city ontology includes 5 object classes, namely networks, blocks, zones, landscapes and focal points (see the previous section). Each object class is a set of object types, each represented as geometry (shapes and parameterised shapes) and attributes (labels). The ontology defines a dependency structure for all the shape sets and label sets that compose an urban plan, enabling grammars to consider them as

parameterised shapes and labels for the application of rules to generate urban designs. Each set of parameterised shapes that is part of the ontology is noted with S_i, in which the index *i* defines the position of the set in the ontology from 1, 2,..., *n* and *n* is the total number of shape sets in the ontology (see class numbers in Figure 9).

As a broad definition, an urban induction pattern is a recurrent urban design move encoded into a small grammar γ that can be applied to replicate the design move in various contexts, and an urban grammar Γ is a specific arrangement of UIPs, i.e. a compound grammar of urban induction patterns.

The model proposed here is shown in diagram form in Figure 11. The UIPs were defined on the basis of the four case studies shown in Figure 7 and the design moves explained and described by their designers. This simplified framework provides a manageable working design domain that is broad enough to develop proof of concept. However, the domain can be extended by (1) enlarging the city ontology and (2) increasing the set of available UIPs. Such a design model provides an ever extendible design model, or a potentially infinite urban pattern grammar – termed in Figure 11 the 'universal' urban pattern grammar Γ_∞ .

The generation module contains all the urban grammars that can be defined through all the possible arrangements of UIPs and all their parameter variations. The urban pattern grammar set Γ is a very generic grammar containing all the specific urban grammars Γ' that can be defined by the Cartesian product of subsets of the UIPs available in the generation module (Figure 11). In short, Γ grammars denote urban pattern grammars or urban grammars made up of patterns, and γ grammars denote urban induction patterns or design moves.

Formally, an urban grammar Γ' is the Cartesian product of user-selected grammars $\gamma_1 \times \gamma_2 \times \gamma_3 \times ... \times \gamma_n$ that take a set of parameterised shapes from the city ontology, $S_1, S_2, S_3, ..., S_n$, respectively, to design an urban plan. The urban grammar Γ' is a subset of the Cartesian product of all grammars γ . A complete layout of an urban plan is defined in 4 design phases which produce 4 sub-designs with different levels of detail. γ grammars are applied in parallel to generate layered representations. Each design phase uses some of the parallel grammars, γ_1 to γ_n of an urban grammar Γ' to generate the several layers that define the sub-design produced in that design phase. Label sets $L_1, L_2, L_3, ..., L_n$, are the label sets in grammars $\gamma_1 \times \gamma_2 \times \gamma_3 \times ... \times \gamma_n$, respectively, and they correspond to the classes of attributes in the ontology. The structure is similar to the one presented by Li for the Yingzao Fashi grammar (Li, 2001) but enhanced with discursive grammars (Duarte, 2001), as suggested by Knight in (2003).

Any urban grammar Γ' is built up from a sequence of UIPs. It is a sequence of design decisions which in the end reflects the design language of the urban plan(ner) and any other decision-makers involved. The same urban grammar Γ' can be used to produce different instantiations by applying different parameter values to the parametric shape rules or by applying the rules in different ways. The urban pattern grammar Γ is, in fact, an algorithmic implementation of part of a Pattern Language as Alexander conceived it, but produced in a way that allows a designer to define his own pattern language Γ'.

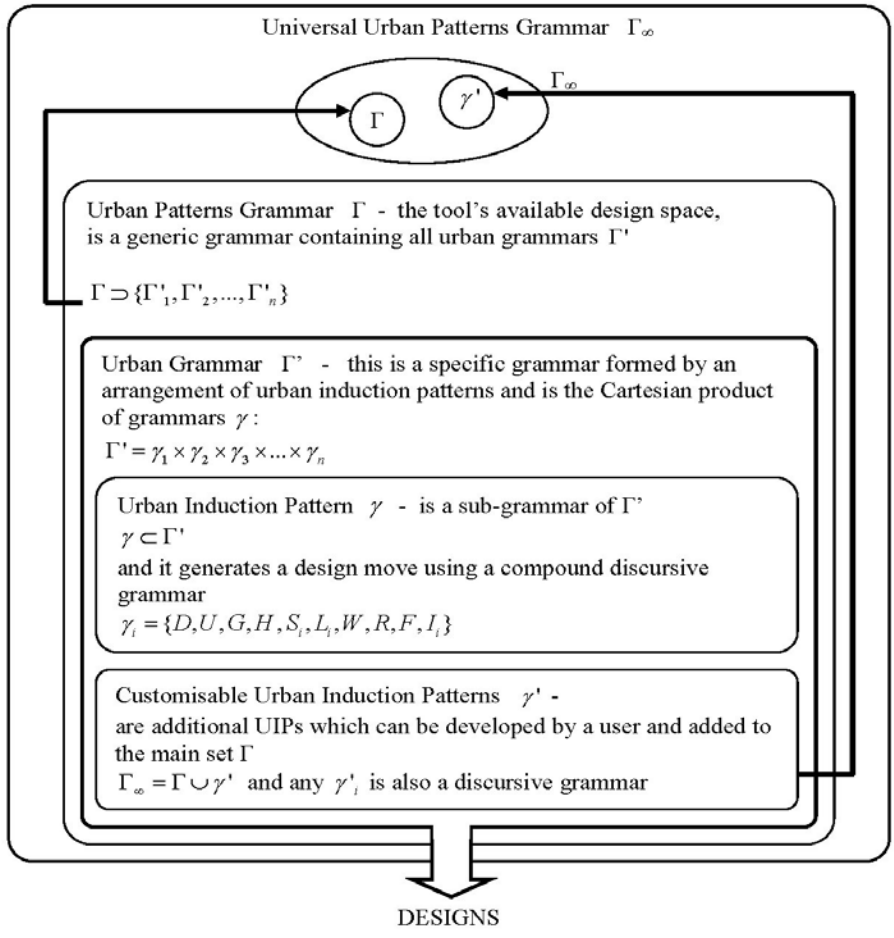


Figure 11
Definitions

Following the previous definitions, a UIP is a sub-grammar of Γ'. A UIP uses some of the parallel grammars in Γ' which are a subset γ of Γ', namely some components of $\{\gamma_1, \gamma_2, \gamma_3, ..., \gamma_n\}$. A UIP is a compound grammar γ composed of a set of parallel discursive grammars γ_i of the form $\gamma_i = \{D, U, G, H, S_i, L_i, W, R, F, I_i\}$ in which S_i is the set of parameterised shapes corresponding to the i^{th} shape object class in the ontology, L_i is the set of labels corresponding to the i^{th} attribute object class in the ontology and I_i is the initial shape. The initial shape I_i is always a shape in S_i generated by a previous UIP or a shape in I_0 in the case of initial UIPs where I_0 is the set of initial labelled shapes. These initial labelled shapes are existing objects found in the representations of the existing context used by the initial UIPs to start the design. There are only two types of initial shapes: I_s, denotes the intervention site limit and R_{ef} objects are labelled shapes

representing selected elements from the site context. The R_{ef} objects are selected by the designer as referential elements in the design context and therefore the best candidates for defining the main guidelines of a plan. Each UIP addresses a goal *G*, which is to be achieved through a set of description rules *D*, starting from an initial description *U*. A set of heuristics *H* decides which of the rules in the set of rules *R* to apply at each stage of the design process. *W* is a set of weights and *F* a set of functions used to constrain the generation to comply with existing regulations and quality standards. Weights may also be used as a way of expressing the relative importance of elements in the design, allowing rules to be applied according to this relative importance.

To sum up, the universal urban grammar Γ_∞ represents the scope of tool's application. UIPs are available and customisable generic design moves expressed through the γ grammars. The available set of UIPs can be extended by the designer by adding a new, customisable grammar γ' to complete the grammar Γ', thereby implicitly extending the grammar Γ towards Γ_∞. Every generic design move or UIP can be further customised by the designer by constraining the range of parameters to be assigned to the rules defining the specific design moves which s/he considers most appropriate for the specific needs of the context. Any arrangement of specific design moves defines a specific grammar Γ' corresponding to a designer's language, which instantiates urban designs within the designer's language through a specific assignment of parameters. The urban pattern grammar Γ is a very generic grammar containing all the possible urban Γ' grammars composed of arrangements of generic design moves (γ and γ'), and corresponds to the available scope of the design tool. The universal urban grammar Γ_∞ can always be extended by (1) adding new objects to the ontology and (2) creating new design moves (UIPs) operating with objects from the new object classes. The discursive grammar structure guarantees that each move follows Schön's reflective structure, which corresponds to a formulation-generation-evaluation structure (see-move-see, in Schön's words).
The distinction between the City Induction model and the UIP model should be made clear at this point. The City Induction model contains a formulation-generation-evaluation structure in which the formulation sets an urban programme (a set of generic goals to be achieved by the final design), whilst a UIP, being a discursive grammar, contains a formulation-generation-evaluation structure that is locally applied if a set of circumstances is recognised in the design context for which a local set of goals is established for the design move. See(1) in the see(1)-move-see(2) cycle stands for: recognise context and formulate local goals. Move stands for: generate design move towards local goals. The generation follows the goals using specific heuristics. See(2) stands for: analyse generated move for validation.
The reader should be aware that this system was defined to produce representations of urban environments in a format that can be interpreted by a GIS platform. As such, the designs obtained are 2D layered representations of urban spaces. This particular detail introduces an important change to how shape grammar representations are usually treated. The maximal shape representation that allows for visual reasoning with

superimposed representations dealing with, for instance, two partially overlapping lines as a single line, can only be considered in each layered representation but never in combination involving lines from different layers. This is because GIS representations offer different overlapping thematic representations to describe the urban environment. For instance, a block representation can contain several plots which may coincide with buildings. In a geographic information system each of these representations is represented by closed polygons and separated in thematic layers, even if visually referring to the same polygon. This structure allows particular kinds of data to be associated with each kind of thematic representation. Moreover, in a GIS, each layer is either composed of points, lines or polygons. No mixed representations are allowed, in order to preserve the topological relationships between the thematic representations. However, despite this divergence from the traditional approach to shape grammars, the fact is that the rigorous formalism it provides allows precise types of representations to be generated and to fit GIS requirements. The parallel grammar structure always allows UIPs to generate representations per thematic layer in the correct format (points, lines or polygons) and with the associated data: axes, for instance, are generated with the associated hierarchy (a_1, a_2, a_3 or a_4). Streets can have associated street names and street descriptions, and blocks can have density regulations associated with them. This characteristic of the design system is what makes it possible to integrate the generation process with the GIS analytical tasks. Nevertheless, it should be stressed that full use of the potential of such integration may only be achieved by users with combined expertise in both CAD (design) and GIS (analysis), two tasks that have up to now essentially been separate and performed by different expert teams[14].

14 Many experts as well as stakeholders would argue that analysis, design and evaluation should always be separate and independent processes. The argument here is not that the urban design process should be integrated into a single process guided by a designer, but that it should instead provide the designer with tools that allow him/her to have upstream and downstream information on the design in order to gather as much information as possible to support design decisions. The fact that a tool can cover the overall process does not imply a single procedure; on the contrary, procedures should be separate but share as much information as possible.

§ 5.7 Conclusion

This chapter defines the theoretical structure of CItyMaker – a design generation model for designing urban plans.

The main scientific contributions of the model defined in this chapter are threefold:

1. The definition of a generative formalism which produces designs (representations + data) in formats suitable for GIS integration.
2. The definition of a relational structure (an ontology) of urban concepts in terms of conventional design reasoning, that is, considering the role of concepts in the design process.
3. Finally, and perhaps the most important contribution, the definition of a shape grammar formalism which allows for design synthesis without the need to predefine a design language. In fact, the proposed formalism allows the language and the design to be defined gradually (and reflectively). The design provides an illustrative solution and the language provides the solution space or, in other words, the flexibility of the design.

These three contributions provide a design system which respects the typical reflective structure of the design process, manages semantic data, and generates representations for geographic information systems. The next chapter will present the methodological approach to using such a design system and the technical details of the urban induction patterns, showing some of their applications and examining the details of their discursive structure.

6 Designing with Urban Patterns

The Urban Induction Pattern (UIP) formalism allows an algorithmic urban design system to be designed based on a progressive combination of typical design moves from which designers can build up their designs reflectively. The gradual definition of a design also involves the definition of the rules needed to generate the design, forming a pattern language in which patterns are design moves. The language also defines the flexibility space of the design. The main concepts described in the previous chapter now need to be examined in further detail to show:

- The UIPs that compose the design system;
- The design method for using them;
- The method for using CItyMaker and the UIP library available in the system;
- That UIPs generative behaviour is based on discursive grammars, that is, shape grammars, description grammars and heuristics;
- That this structure allows semantically sound designs to be generated;
- That the system's options (UIP selection, optional rules and parameter manipulation) allow for a large design exploration space. This will be illustrated with some examples – the UIPs used to generate the main case study, its derivation and the design variations that the system's options allow;
- That the UIPs defined allow for the generation of the plans from which they were inferred as well as the generation of other design solutions;
- How UIPs make use of the concepts in the ontology by showing some details of their representation.

§ 6.1 Using Urban Induction Patterns – a methodological approach

In CItyMaker, the design language is synthesised throughout the design process through the progressive selection of UIPs, selecting their optional rules and constraining their rule parameters to customised values. A set of fixed constraints defines limits for the system components and rule applications within the prescriptions of regulations and quality standards. This set of constraints is defined a priori by the designer by introducing the applicable local regulations and standards. In the City Induction project, this information is managed in the ontology.

In order to establish the main structure for the implementation of CItyMaker a wide set of UIPs was defined, based on the 4 urban plans used as case studies (Figure 7). Each UIP captured the rules underlying the designers' moves when designing their respective plans. The UIPs were codified very abstractly using a discursive grammar in such a way that they could produce different formalisations of the same basic move just by setting different values for the available rule parameters. In some cases the UIPs have other open options or, in other words, alternative subsets of rules that follow a common initial set of rules and a common concept. These subsets produce alternative solutions to the main problem addressed in the pattern. If no restriction or constraint is previously defined by the formulation module, the decision is up to the designer, who is therefore allowed to follow his/her design convictions. These optional subsets of rules within UIPs are evident in Table 6 in UIPs such as *MainAxis* or *DefineUUnit* where three optional output types can be chosen.

The verification of this concept can be tested by checking the UIP application in the different plans. Each pattern can be used to explain the generation of certain design features in some of the case studies and even in other well-known urban plans such as Cerdá's Barcelona plan (see Table 14 – page 196). Design diversity can be achieved by combining different design moves and manipulating the available parameter values. The main idea is that most urban designs are the result of different combinations of the same generic design moves.

Table 6 shows the list of UIPs developed during the current research. The application sequence is not predetermined, but equally is not random. It is always based on the properties of the previous design iteration, which are recognised by the UIPs whether based on shape recognition or attribute (label) recognition. Whatever the sequence may be, several design levels are identified and separated by grey rows in which the main goals of these design levels are described. The icons in the table show the UIPs that have been, or are being, implemented as an AutoCad extension (Beirão et al., 2010). The system can always be extended, although the amount of UIPs developed from the case studies is diverse enough to produce a very large number of design variations.

In very general terms a design follows several cycles of formulation, generation and evaluation. Each of these cycles is preceded by an analytical phase and followed by some evaluation procedures (visual, data-based or tool-based evaluation). Design cycles usually end when the requirements of a design phase are completed. In detail, the regular design cycle workflow is based on the following sequence of procedures:

1. context analysis – gathering and structuring data from the design context using GIS as the analytical platform (formulation)
2. programme formulation – setting the goal specifications for the plan (formulation)
3. preparation of the drawing base (the design generation base or initial design features) – the generation process starts with the definition of the intervention site, I_s, represented as a polygon inside which the designs will be generated (as a manual procedure or imported from the GIS shape file – generation)

4 start – a kind of 'Start Design' button (generation) – prompts a table for checking the list of specifications defined by the formulation and allows the user to further customise, change or check the available parameters
5 reference (R_{ef}) selection – the selection of pre-existing elements that the designer wants to use as referential items to support the design guidelines (manual selection and automatic labelling of pre-existent elements – generation)
6 UIP selection and generation (generation) – selection of design moves from a list of available UIPs and automatic generation until they prompt a request for:
7 parameter assignments or optional rules – if these parameter assignments or optional rules have not already been defined by the formulation module
8 looping back to redefine previous decisions or jumping to the next step – loop back to start (point 4)
9 end of generation, with a plan layout output
10 evaluation procedures – checking the design resulting from the generation process and feeding back information on the quality of the results (evaluation)
11 looping back to programme formulation or any point in the generation process to redefine designs to meet redefined goals.

Urban Induction Patterns – name ▼		Urban Induction Patterns – short description ▼
A . Creating the Composition guidelines (part of phase 1)		
001- MainAxis 	*MainAxistheLongerLine* *Cardus* *MIAxis* (Most Important Axis)	*MainAxis* generates the main axis of the urban design composition and has 3 options: *MainAxistheLongerLine / Cardus / MIAxis*. The latter is the Most Important Axis which is an axis that passes through the two most important R_{ef} entities. The *Cardus* pattern may only be applied once.
002- OrthogonalAxis (Decumanus) 	*OrthobyMidPoint* *OrthobyLongerLine* *OrthobyMIAxis* (Most Important Axis)	Generates an axis passing through a R_{ef} point which is orthogonal to any previously generated axis or to an existing axis. It can be applied as long as there are still R_{ef} entities to use. If the existing axis is a *Cardus* the generated axis will be a *Decumanus*. A *Decumanus* can be generated only once.
003- Promenade 		Transforms an existing axis into a *promenade*. Adds components to the street composition.
004- CompositionAxis 		Generates a composition axis passing through one or two R_{ef} points or as an extension of a R_{ef} segment. It corresponds to other applications of *MainAxis* without the *Cardus* option. It can be applied as long as there are still R_{ef} entities to use. It can only be used after *MainAxis*.
005- AStA (Assign Street type to Axis)		Assigns a particular street type to an axis attributing labels. The classification will later allow for the detailed definition of street properties.
006- PZSubdiv (Property Zoning Subdivision)		Property and zoning subdivision: determines all property subdivisions until block definitions are reached – urban plots surrounded by streets.
007- RingRoads		Draws a road circulating around a bounded area.

008- InsertFocalPoint 	Inserts a focal point. A point + the attribute F_p (focal point). Focal points can be placed in R_{ef} s inside the U_a area or in places with privileged views. The user can also choose to manually position an extra focal point.
B . Creating Grids (completing the street network) (part of phase 2)	
009- (grid by) *AddingAxes* 	Picks two orthogonal axes or two intersecting axes and offsets new axes (a_4 axes) until a grid is completed inside the urbanizable area, U_a. If there are still R_{ef} s or focal points available there is an option to generate a higher hierarchy street instead of just a_4 axes.
010- (grid by) *AddingBlockCells* 	Picks two orthogonal axes and adds block cells recursively until a bounded space is filled. This pattern contains some adjustment patterns that apply only if needed, usually due to greater variation in the parameter values.
011- (grid by) *RectangleDissection*	Recursive subdivision of a zone until a stopping condition is reached. The stopping condition is a minimum block area. Another parameter constrains the rectangle proportion in the subdivision process.
012- IcerayGrid	A similar recursive subdivision, but the subdivision may not be orthogonal. It follows rules similar to those of Stiny's iceray grammar (Stiny, 1977). The angle of subdivision becomes an extra control parameter.
013- RadialGrid	Generates a radial grid from a focal point creating a plaza at the centre. It can only be applied if a focal point has already been used.
014- AlignStreets	Used to correct generation maladjustments in orthogonal grids. Aligns streets connecting to another street if their intersection points are closer than a certain predefined distance. The pattern applies automatically after patterns *010* or *011* are applied.
015- ConnectStreets	Corrects unfinished generation details using patterns *010* or *011* to connect streets left unconnected. This pattern applies automatically when patterns *010* or *011* are applied.
C . Transformations to the street network (part of phase 2)	
016- AxisOverGrid 	Applies a new axis over a grid. This pattern contains adjustment patterns that are applied to solve some of the incongruous resulting shapes.
017- RotateAxisbyNode 	Rotates an existing axis towards a R_{ef} point using a node as rotation centre.
018- MoveGridNode 	Moves a grid node to a new position according to specified parameters.
019- ChangeStreetSegment	Moves a street segment to a new position or erases it.
D . Creating Public Space (available in phases 1 to 3)	
020- AddPlaza 	Places a marker in a two main street crossing or in a R_{ef} point inside the U_a area. The *GeneratePlaza* pattern will replace the marker with a specific plaza type (always a *Sq1* type).

021- GeneratePlaza 		Replaces a plaza marker with a *Sq1* square type.
022- InsertPublicSpace 	*Sq2*	Inserts a public space (that is, a *Sq2*, *Sq3*, *Sq4* or *Sq5* square type) in a focal point or a R_{ef} point or adds *Sq2*, *Sq3*, *Sq4* or *Sq5* labels to blocks or junctions in the case of *Sq4*.
	Sq3	
	Sq4	
	Sq5	
023- Square 	*SquarefromBlockSubtraction* (*Sq2*)	Subtracts an *island* or a block unit and places an abstract square with the same geometry as the *island*.
	SquarefromBlockTrim (*Sq3*)	Reduces the length of the longer side of the block creating a square (*Sq4*).
	SquarefromCornerTrim (*Sq4*)	Trims the corners of blocks in a junction creating a small square. It can be used to create chamfers in corners, as in Barcelona.
	CutPublicSpaceinBlock (*Sq5*)	Opens holes / subtracts parts of the *island*. Generates blocks similar to those found in the IJburg plan.
E . Creating Urban Units (part of phase 3)		
024- AddBlocktoCells 		Adds an *island* (or urban unit) to a cell defined by a set of axes. A cell or block cell is a closed area defined by a closed set of street axes or by axes and part of a boundary of a zone (usually a U_a).
025- AdjustingBlockCells 		Adjusts blocks overlapping the boundaries of the intervention site or urbanisable area. Several operations are allowed to adjust designs along the border areas of the design.
026- ClassifyUUnitCells 		According to their location within the intervention site and relationship to reference elements, some cells may be more suitable for certain block types than others. In addition, the designer's style may be expressed by playing with different alternative block types. The UIP creates a label in each cell that states which block type or types are allowed to be created in that particular cell.
*027- DefineUUnit ** 	*Neighbourhood*	This is a rather complex pattern with an extensive algorithm. It defines a set of requirements needed for a minimum neighbourhood definition, sets the values for the available parameters and generates the neighbourhood parametric block with attributes to be used in bottom-up generation. An extensive list of specifications for a neighbourhood programme is set here if no previous definitions were set by the formulation module.
	BlockType	Defines a block type, sets the parameters and range of their variation and creates a block type attribute as an identifier to use in generation.
	Cluster	Defines a cluster of buildings and/or private or public spaces, their parameters and relationships expressed through the range set for their accepted values, attributes, and the definition of a bounding box to be used later in the generation.
028- InitialUUnit 		Places an urban unit in a R_{ef} or a Focal Point. Creates the initial labels for the recursive application of *AddingUUnits*.

029- AddUUnitbyLabel 	*AddBlockType* 	Replaces an *island* with a block type.
	AddCluster 	Replaces an *island* with a cluster of buildings.
030- AddingUUnits	*AddingNeighbourhoods* (BU**)	Recursive addition of neighbourhood units. Some rules create new labels to maintain generation; others erase the labels creating terminal conditions to stop generation.
	AddingBlockTypes (BU) 	Recursive addition of block type units. Some rules create new labels to maintain generation; others erase the labels creating terminal conditions to stop generation.
	AddingClusters (BU) 	Recursive addition of cluster units. Some rules create new labels to maintain generation; others erase the labels creating terminal conditions to stop generation.
031- NeighbourhoodGeneration 		The algorithm starts by identifying the potential areas for neighbourhood definition according to a predefined range of values for the neighbourhood surface and a set of constraints for boundary definition, identifies the existing features in each area (blocks, functions, streets, public spaces) and, depending on the context, makes the necessary adjustments to fit the required neighbourhood specifications. The final part actually performs several alternative routines depending on the context and specifications. The algorithm is rather complex and contains many subroutines from other patterns, such as *AssignFunctions*, *InsertPublicBuilding* or *Square*.
032- InsertPublicBuilding (public facilities) 		Inserts a public building in a Focal Point or in the centre of an activity zone.*** Alternatively, the designer is prompted to define a location.
033- InsertBuilding 		Inserts a single building in a focal point.
034- BuildingHeadingAxis		Inserts a single building in a position heading an existing axis. The algorithm breaks the existing axis, adds a hierarchic classification to both building and axes, and places a focal point.
035- AddArches		Adds an arch to a street. It can be applied at the end of two blocks assuming the appearance of an entrance, in the middle or at the centre of a square using one of the axes for orientation.
F . Others – detailing patterns and managing functions (available in phases 3 to 4)		
036- AddAccessStreet 		Adds a local access street.
037- AssignFunctions 		Assigns functions to blocks, buildings or floors or changes the functions of blocks, buildings or floors.
038- ManageBuildingHeight 		Assigns or changes building height in terms of number of floors. Involves managing the following parameters: average number of floors (*L*), maximum number of floors allowed (*H*), maximum height allowed ($_{max}H_{ei}$), ground floor height (G_{roFH}), floor height for all floors or per floor (F_{HeI}) and (F_{HeiFi})

039- JunctionTypeMarkers	Creates a marker identifying a junction type. This is automatically generated from the street classification.
040- SubdivideBlockintoBuildings	Subdivides a block type into buildings. The algorithm defines building regulations.
041- SubdivideBlockPropertyintoPlots	Subdivides a block type into plots. The algorithm defines plot regulations.
042- StreetTypeComposition	Manages the kind of street components used to compose a particular street type. See section § 5.4 in the thesis.
043- StreetTypeDesign	Designs the street according to predefined compositions previously attributed to street types.
044- JunctionType	Defines the junction type conjoining street types.
045- JunctionTypeDesign	Replaces the marker with a particular *JunctionType* design.
UIPs for managing zone and landscape design are not defined here, nor most of the detailing level patterns.	

* *DefineUUnit* is a complex pattern. It has 3 options that are immediately prompted when chosen: *Neighbourhood*, *BlockType* and *Cluster*. *BlockType* can be used before (bottom-up) or after (top-down) the grid generation. The same applies to *Cluster*, but *Neighbourhood* as defined here is for bottom-up generation only. Another UIP, *NeighbourhoodGeneration*, defines neighbourhood requirements and adapts the features required to define a neighbourhood to the grid. The algorithm is different, although it pursues similar goals. Note that *DefineUUnit* does not generate units, but simply defines a base to be replaced later with other UIPs such as *AddBlockType*.

** BU – Bottom-Up

*** This concept is not explained here in detail and is still under development, but to keep the idea meaningful it may be said that each entity in the plan has a degree of attraction or repulsion to an activity (social activity). Activity areas can be mapped following the integration concept from space syntax (Hillier and Hanson, 1984). This is one of the concepts that need further integration into the work developed within the City Induction project.

Table 6
Table of urban induction patterns and brief description of their algorithms.

These steps require further exploration.

1/2 These two points concern the formulation module. However, the designer is supposed to have a list of specifications for the planning goals, for example, an Excel sheet with the necessary data derived from some analysis of the existing context. The designer is supposed to be able to fill in this information in the checklist mentioned in Point 4. The necessary information is a set of requirements or goal indicators which constitute the urban programme. The urban programme may be developed using traditional media or the tools developed by the formulation module. Depending on the availability of data, the process can be manual or semi-automated. A set of actors may be involved in filling in the programme requirements.

3 The drawing preparation can be a manual or semi-automated procedure depending on the tools available upstream in the formulation / analysis process and also on the specific characteristics of the design context. For instance, the intervention site, I_s, may be constrained by higher level regulations or topography, and therefore may not allow for construction on the entire intervention site

surface. The resulting area available for construction can be called the urbanizable area, U_a. However, network connections (streets) may need to link isolated U_a areas. Thus, some rules apply to I_s whilst others are operational only at U_as level. The main axes patterns, for instance, which are responsible for determining the design guidelines for the main elements of the street network, may apply to I_s, whereas grid generation and block generation patterns may only apply to U_as. I_s and U_as are defined in different design layers and the UIP rules respond differently to them.

4 The 'Start Button' prompts a list of specifications to be fulfilled. The list may or may not be completed in its entirety. However, minimum requirements are mandatory; otherwise the list is returned when checked. During the generation process the UIPs seek information on the specifications available in this list. If some specific information is not available, the UIP prompts another request before continuing. There are 2 main types of specifications that may be needed for generation; firstly, generic specifications and goal indications (Table 7) concerning the urban parameters at district level and, secondly, design specifications concerning design components, such as street components or block size input specifications. For the street network generation, for instance, the system will need the street width input for each street hierarchy. For block generation it will need the maximum and minimum block size, maximum and minimum building depth, and maximum and minimum courtyard depth (Table 8). Some generation goals can be set in terms of density indicators to be fulfilled. For instance, in some contexts the pre-existing general regulations might predetermine density goals set as maximum building intensity, whilst in other contexts they might be set as maximum coverage and maximum number of floors. Given that interdependent fields are linked by their mathematical relations, the designer only needs to fill in the known information and the linked fields will be automatically filled in. This provides the designer with new data on related factors. Nevertheless, the designer can change parameters at will, providing freedom to explore the design. Spacematrix (Berghauser-Pont and Haupt, 2010) density measures play a fundamental role here because they provide the model for calculating the density measures accurately.

5 References (R_{ef}) are points, lines or polygons selected from among pre-existing ones to act as referential elements for the design. The initial rules of the initial patterns available take these R_{ef} entities as their initial shapes for the generation of designs. Some algorithms can be developed to prompt suggestions for R_{ef}s. However, because this procedure implies semantic interpretation of contexts involving great subjectivity it was decided that this should remain manual[15].

15 Subjectivity here refers to common designer idiosyncrasies, what Janssen calls designer preconceptions (2006) and Minsky calls default assumptions (1988). This kind of subjectivity contains two main characteristics: the designer's personal approach to design and tacit knowledge.

Nevertheless, some UIPs can create new R_{ef}s although most of them erase the used R_{ef}s. R_{ef}s can be classified with different weights so that the rules can perform qualitative decisions distinguishing between very important references and less important ones. The rating criteria can be used by some UIPs to determine how the rules are used (e.g. the *MIAxis* option in *MainAxis* – Table 6). Weighting is essentially a manual procedure introducing a high level of subjectivity through designer interpretation.

6/7 The UIPs search for parameter inputs in the list of specifications mentioned in Point 4. If a required specification is not found, the UIP prompts a request to fill in this information. The system was designed to allow programme information to be filled in at different stages of the design generation. As such, the design process becomes a more interactive but also a more reflective process. This characteristic generally reflects the existing literature on the design process (Schön, 1983); (Lawson, 2006); (de Jong, 2009). The generation system produces a record of the sequence of UIPs chosen, their options and the parameters applied, which can be used later for simple consultation or for re-evaluating individual design options.

8 This is a design loop, not exactly a feedback loop but a reflective loop, as it allows the designer to rethink his/her decisions. This process should occur at the end of each design level in order to provide a high degree of reflectivity.

9 At a certain point the designer will be satisfied with a preliminary design output resulting from the generation sequence and be ready to evaluate the results in order to refine or even rethink them. This 'output for evaluation' is not necessarily a complete final design. Outputs for evaluation may be taken at the end of each design phase in order to allow for intermediate evaluation procedures. This is also important because usually local authority approval processes are also phased in several steps[16]. The main idea here is to use the relationships defined in the ontology to control this behaviour. Relationships between objects can be established on the basis of inherited properties, without which certain operations may not occur or may be needed in consequent steps of the design. In addition, a minimum set of specifications might be required for each design phase. For instance, a design phase may only be finished when an n number of specifications is fulfilled by generation, but a new phase can only start when a $n+a$ number of specifications is fulfilled, n and a being natural numbers $(n,a \in \bullet \wedge n,a \neq 0)$.

16 Approval procedures are similar in general terms in many countries regarding the levels of detail in each phase. The four design levels referred to on page 104 (Section § 5.3 Case Studies) were taken from (Beirão, 2005) following a study on urban design methods. The four levels indicated here are also shown to be consistent with regular approval procedures in Portugal. The same kind of consistency can be found in the Dutch case studies and a similar level subdivision is clearly found in the Ypenburg and IJburg plans. Beirão also shows the Borneo-Sporenburg case in Amsterdam, which again follows similar decision-making levels.

Urban Parameters and Attributes			
Type of data	**Parameter / indicator / attribute**	**variables**	**Units**
Given data	Intervention Site Area ($I_{s\text{-}area}$)		hect or m^2
	Construction site or urbanizable area (area fit for construction) (U_a)		hect or m^2
Quantities	Number of households (n_h)		Integer
	Maximum Allowed Gross floor area ($_{max}AG_{FA}$)		m^2
	Non-building areas (per type):		
	Protected areas (P_{rotA})		hect or m^2
	Parking areas (P_{arhA})		hect or m^2
	Public Parks and Gardens (G_{ardA})		hect or m^2
	Private outside areas (P_{rivoA})		hect or m^2
	Street areas (S_{treeA})		hect or m^2
	Others public space areas (O_{thpsA})		hect or m^2
	Gross floor area (F_d)	Footprint x nr of floors	m^2
	Maximum Allowed Footprint area ($_{max}AF_A$)	Footprint area	hect or m^2
	Network length (l): (l) $l = l_i + \frac{l_e}{2}$ l_i – length of interior network l_e – length of exterior network		Real (in m)
	Footprint (B_d)	Ground floor area	m^2
Indicators	Building Intensity (FSI)		(ratio) m^2 / m^2
	Density - Households / hectare(D_h)		(ratio) n_h / hect
	Network Density: $N_d = l / l_{s-area}$ (N_d)		m / m^2
	Coverage (GSI)		(ratio) m^2 / m^2
	Spaciousness (OSR)		(ratio) m^2 / m^2
Height	Average number of floors (L)	Above + Underground	Real
	Maximum number of floors (H)	Above + Underground	Integer
	Maximum Height ($_{max}H_{ei}$)		Real
	Ground floor height (G_{roFH})		Real
	Floor height (all floors) (F_{Hei})		Real
	Floor height (per floor) (F_{HeiFl})		Real
Uses	Ground floor uses – mixed (Ug)	Residential Commerce Offices Services and facilities Industry Others	% % % % % %
	Upper floor uses – mixed (Uu)	Idem	%

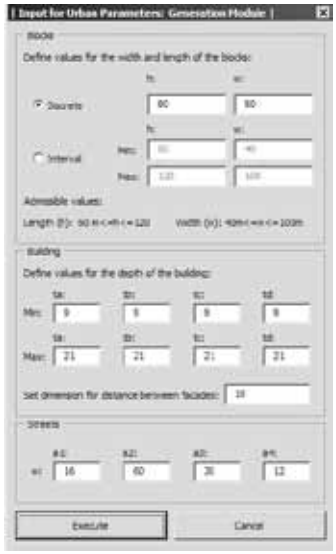
	Exterior uses	(*Eu*)	Public space (total)	%
			Private space (total)	%
			Street areas	%
			Parking spaces	%
			Green Areas	%
Cost	Requires specific model			
Sustainability criteria	Requires specific model – this is being developed together with the formulation model (Montenegro, 2010).			

Table 7
Table of urban parameters and attributes – district scale

10/11 These two points refer to evaluation procedures. The evaluation procedures maybe performed visually, based on the existing data provided by the generation system, or by using existing evaluation tools usually based on GIS platforms (e.g. *space syntax* (Hillier, 1996) / *place syntax* (Stahle, Marcus, and Karlström, 2007)). In the context of the City Induction project, specific evaluation tools will be provided by the evaluation model. However, Point 10 concerns the real feedback loop where the information produced by the evaluation returns to formulation and generation in order to fine-tune the goals and designs. This is the point at which optimisation algorithms might be introduced into the system. Optimisation algorithms may be applied to a record of the sequence of UIPs, options and parameters, i.e. to a memory of all the steps in the generation of the design, and restructured according to the results of the evaluation.

The above description can be considered a rough sketch of the design method for using the proposed urban design system and outlines its embedded algorithm.
CItyMaker needs a set of databases to manage the data involved in the generation process. These data structures will be part of the City Induction ontology common to the three modules (formulation, generation and evaluation). The generation module, CItyMaker, deals with three kinds of databases: (1) existing data – on context, existing regulations and design standards; (2) goals and requirements; and (3) generated data – the data being generated step by step by the generation process, including a register of the design sequence (defined in terms of patterns, optional rules and parameters), and an update on the existing data.

Input Urban Parameters / Control Parameters



Type of parameter	Parameter / indicator / attribute	Variables	Units
Street parameters	Street hierarchies (definition of a range of values for street width for the complete set of street hierarchies – a_1; a_2; a_3; a_4)	Street width	Real (in m)
Block size variation	Range for parameters *h* and *w*. Discrete value or an interval for *h* and *w*	< *h* , *w* >	Integer
Building depth	Defines minimum and maximum building depth to control block generation	< t_a , t_b , t_c , t_d >	Real (in m)
Courtyard depth	Defines minimum and maximum courtyard depth to control block generation	< i_a , i_b >	Real (in m)

Table 8
User input parameters / control parameters – the image shows the window prompt already implemented in the prototype implementation; courtyard depth is not implemented in this window.

An urban design is completed by generating solutions from a complete set of UIPs until the final required level of detail is achieved. A complete set of UIPs is defined as a set including UIPs for all four design levels. Each UIP follows an algorithm, i.e. a set of instructions to generate a typical design move. Table 6 gives a short description of what the patterns do. The developed patterns/algorithms are sufficient to provide an idea of how a complete design is reached. The solution space is limited, bounded by the design languages of the 4 case studies. However, the universe of design solutions provided by this set of UIPs is already way beyond the capacity of human prediction. Moreover, the system can be extended by defining new UIPs or by extending the ontology supporting the design system.

§ 6.2 Generating urban designs with Urban Induction Patterns

This section intends to demonstrate that the concept described above can be used to design urban plans and explore design spaces. Since it would be impossible to start with the aim of defining all possible urban design moves, the work focused on capturing UIPs only within the design space defined by the 4 case studies (Figure 7), attempting to define them in such generic terms that each UIP could be used as broadly as possible regardless of context. The UIPs were defined as far as possible as the smallest definable design moves so that most of the large design moves would already be a compilation of smaller UIPs. In fact, the analysis of the case studies demonstrated that although all the grids were quite different in terms of the final results, most of them were actually obtained from different arrangements of a minimum number of UIPs designed for this purpose.
This section explains the approach used to infer UIPs, provides a summary of the UIPs inferred from the analysis and shows how these UIPs can be used (1) to replicate the case studies, (2) to generate alternative solutions by applying different instantiations of rule parameters and (3) to generate various designs for different contexts.

§ 6.2.1 The common structure of Urban Induction Patterns

Urban Induction Patterns compute objects found in the ontology (see Section § 5.4 , page 92). Objects have a specific meaning that can be understood by the urban designer. They represent meaningful parts of the city. These objects are used by CItyMaker as shapes manipulated by shape grammars or descriptions of concepts and designs manipulated by description grammars. The definition of a common structure for objects allows clear relationships to be defined between the elements in the ontology and for the generation module to recognise them, their meaning and, therefore, their semantic role in the shape rules.
UIPs are based on Gamma et al's design patterns (1995) and in basic terms correspond to generative urban design moves. For the sake of clarity, UIPs have a common structure composed of 10 parts, as explained below. However, the full UIP structure is extensive and would result in a very large document if this thesis were to include the complete list of UIPs described in detail according to the complete standard structure. To avoid this, Appendix 2 provides a library of urban induction patterns in 3 formats:

1 a table of all the UIPs developed during this research, including only a short description of the embedded algorithm;
2 the complete structure (including the 10 parts);
3 the simplified structure.

1 - Table

The table contains all the UIPs that were inferred from the case studies or from variations of the same basic design moves which may be found in recurrent design situations. Not all the described moves were developed into a complete standard UIP format. Some are simply a short description of a common design move observed within the design space of the case studies or other paradigmatic urban plans. Others had only outlined some shape grammars and never achieved the complete discursive grammar format. A few were developed in detail consist with the formal structure of UIPs. The table shows only the short descriptions, but provides a brief overview of the whole design system to facilitate an understanding of how this works. The table includes also an icon corresponding to the UIPs implemented in AutoCAD.

2 - Complete structure

Formally, an urban pattern is an adaption of the Alexander et al (1977) and Gamma et al (1995) concepts. Gamma et al propose that patterns should be composed of 13 parts: *Name* / *Intent* / *Also Known As* / *Motivation* / *Applicability* / *Structure* / *Participants* / *Collaborations* / *Consequences* / *Implementation* / *Sample Code* / *Known Uses* / *Related Patterns*. It is recommended that their online book is consulted for detailed definitions (http://www.whigg.cas.cn/resource/program/CPP/201010/P020101022562155422801.pdf): the definitions which follow indicate only the proposed adaptations.
An urban induction pattern is composed of 10 parts: *Name* / *Intent* / *Also Known As* / *Known Uses* / *Description* / *Structure* / *Predicate* / *Consequent* / *Discursive Grammar* / *Related Patterns*.
Name / *Intent* / *Also Known As* correspond to Gamma's definition. *Known Uses* appears here because it corresponds, to a certain extent, to Alexander's archetypal illustration, to which we have only added an explanatory caption. *Description* corresponds to a simplified description of the main algorithm and replaces *Motivation* / *Applicability*. *Structure* includes a graphic representation of the classes used in the pattern and a pseudo algorithm defining how to generate variations of the pattern. *Predicate* defines the initial shapes and the objects used in the pattern, their relationships and parameters. These are the predicate conditions of the UIP and they replace *Participants*. *Consequent* corresponds to a description of expected results. The Predicate therefore contains the descriptions of the participant objects and active attributes which are used to define the matching function in the rules. Gamma's *Consequences* / *Implementation* / *Sample Code* are replaced by a *Discursive Grammar*. The discursive grammar encodes the shape descriptions, shape rules, and parameters used to generate the pattern. It also contains additional heuristics to search for the

best rule application in each iteration. *Related Patterns* follows the same logic as Alexander and Gamma, but in this case *Related Patterns* can be defined as the patterns that use objects belonging to an instantiated pattern as initial shapes. *Related Patterns* are constrained by the sublevels included in the *IDCode* of objects. These codes define the dependency structure and are managed through the ontology in which all these concepts and relations are stored. All instances in a design have an *IDCode* which has the following format: *< ObjectType, ObjectNumber, ClassName, Sublevels >* in which *ObjectType* is a particular type of instance of a class, *ObjectNumber* is a counter or object identifier (integer), *ObjectClass* is taken from the ontology classification and *Sublevels* indicates the sublevel inheritance properties. *Sublevels* are the future available parts of predicates. *Sublevels* may indicate an object class, meaning that all the objects in that class are available for recognition by other UIPs, or may simply indicate a specific object type, meaning that the type is the only one responding to inheritance. In other words, *Related Patterns* indicate the patterns that are available for a subsequent step in the generation process.

Objects in the ontology have a specific meaning that can be understood by the urban designer. They represent meaningful parts of the city. The objects are used by the generation module as shapes manipulated by shape grammars and descriptions manipulated by description grammars. The definition of a common structure for objects allows for a clear relationship between the elements in the ontology and for the generation module to recognise them, their meaning and, therefore, their role in the shape rules. Objects are defined by a *Geometry*, a set of coordinates assigned in a particular *Coordinate System*. The *Geometry* is a parametrical representation of a city feature with specific parameters and a position defined by the *Coordinate System*. The *Coordinate System* links to a specific coordinate system and enables a particular instantiation of the object to be linked to its representation in a GIS platform.

Objects have *Attributes* and *Properties*. There are two types of attributes: quantitative and qualitative. *Quantitative Attributes* assign data values to objects (e.g., *Volume*, *NumberOfFloors*, *ConstructionArea*, *BuildingAge*, etc.). These values are directly linked to the Geometry. *Qualitative Attributes* assign a qualitative value to the urban entities (e.g., *Function*, *HistoricalBuilding*, *Landmark*, etc) or, in other words, these attributes are labels assigned to shapes. Properties are qualities of shapes that can be assessed during the design process; density measures, for instance, at block level are the properties of blocks. Similar properties can be assigned to zones.

3 - Simplified structure

The simplified structure of an urban induction pattern is composed of 6 parts: *Name* / *Short Description* / *Predicate* / *Consequent* / *Diagram* (illustration) / *Related Patterns*. The *Name* itself, as in Alexander and Gamma's definition, should contain recognisable evidence of what the design move is, at least for an urban designer. The *Short Description* is the same as in the table. The *Predicate* is a simplified description of the predication conditions applicable in the pattern. The *Consequent* is a simplified description of the pattern's solution. The *Diagram* illustrates the main transformation

in a simplified diagram. It is not a formal grammar but a symbolic representation of one. The *Related Patterns* follow the same logic as the previous cases. Most patterns will be represented in this format.

§ 6.2.2 Defining UIPs

All the Urban Induction Patterns defined for CItyMaker produce designs of urban plans in typical plan representations. All the UIP rules are reproduced in Appendix 2 – A Library of Urban Induction Patterns.
As previously stated, the first UIPs applied are the ones that take the intervention site limit I_s (and/or urbanizable areas U_a) and elements selected as references, R_{ef}, as their initial shapes. The first UIP used within the framework of the case studies was suggested by the explanation given by Frits Palmboom, the author of the Ypenburg plan, concerning his design methods. When explaining his first move in the design of the Ypenburg plan he said – "I always look for the longer line in the territory", then showed a few examples of other plans following the same principle. This was clearly a pattern. Architect Chuva Gomes explained the same principle in different words.
Taking this sentence as a byword, the first UIP was called *MainAxisistheLongerLine*. The algorithm generates all the lines that can be produced using all the referential elements (R_{ef}) previously marked by the designer and selects the longest one. It is applied in all the case studies and takes the particular form of the *Cardus* in the case of Plan 1, selecting from the proposed long lines the one with the closest north-south orientation. The term *Cardus*, taken from classical urban planning, already demonstrates the possibility of applications in a wider design space than the one defined by the case studies, namely in the generation of classical plans. Following the first UIP, others were devised. *OrthogonalAxis*, or *Decumanus* if perpendicular to *Cardus*, represents another typical design move. In Plan 2 it can be seen that these two sequential UIPs are used differently in 4 different areas of the site – see Figure 8. All of these are particular cases of *CompositionalAxis*, a UIP that uses two references to draw a compositional axis, and they are all related to the first design level (A). The rules of *Cardus* and *Decumanus* are shown in Appendix 2.
At the second design level (B) we considered two main Urban Induction Patterns to generate the grids and a few UIPs corresponding to complementary tasks that adapt the grid to predefined conditions, adjusting the results along the intervention boundaries or any other already existing element. The two UIPs generate grids by *AddingAxes* or by *AddingBlockCells* and have been used to reproduce the design of Plan 1. The two different UIPs are able to generate Plan 1 because the parameters attributed to block length (h) and width (w) are discrete values (50m x 80m) as defined by Chuva Gomes. However, the two UIPs produce distinctive types of urban grids if the

parameters vary, for instance within a predefined interval at each step in the derivation. The nature of the designs obtained in this way is quite distinctive, as shown in Figure 12. In the case of *AddingBlockCells*, some specific rules were needed in order to deal with the use of varying parameters. During the derivation with *AddingAxes*, third hierarchy axes were generated and adjusted using a set of optional rules (Rules 4c and 6c - see Appendix 2). At the end of the grid derivation, two more UIPs were used – *AddBlocktoCells* and *AdjustingBlockCells*, to create *islands* (empty block structures within the street cells) and adjust blocks to the boundary conditions of the design respectively, adapting the generated grid to the boundaries of the site. In terms of the goal of defining a design tool, the difference between the two grid generation UIPs is viewed as an extremely positive quality because it encourages design exploration.
The generation of a grid by *AddingBlockCells* also has the particular advantage of generating the grid progressively in a bottom-up fashion starting from two orthogonal axes. Bottom-up urban generation deserves some attention as the generative mode is consistent with the natural evolution of organic grids. There is no particular interest in generating grids in a bottom-up fashion if the final result is a layout presentation similar to current practices in a top-down mode except for the morphological exploration of other types of grids. However, if the bottom-up behaviour of the generation can be used as some kind of plan regulation, by which some qualities of the final grid can always be guaranteed through rule constraints embedded in the pattern, then bottom-up patterns definitely acquire a particular interest in the development of urban plans. Moreover, spatial analysis of organic grids (Hillier and Hanson 1984) has shown that there are morphological characteristics embedded in the topological structure of such grids which are responsible for the acclaimed qualities and positive social activity in this kind of urban fabric. Therefore, the morphological exploration of grids based on bottom-up generation alone can be a valid approach to improving the quality of planned grids, although in this case the analysis should be integrated into the synthesis process for validating solutions, due to the difficulty of making human decisions involving visually disorganised solutions as opposed to visually ordered ones.
During the current research some UIPs were developed inspired by bottom-up behaviour (see UIP 030 and options in Appendix 2). Others were inspired by organic-like grids. The *RectangleDissection* (UIP 011) grid is based on a recursive rectangle subdivision and was inspired by Marshall's description of a characteristic route structure (2005)[17] . It is also an orthogonal simplification of *IcerayGrid* (UIP 012), a grid inspired by Stiny's iceray grammar (1977) which had already proved efficient in replicating traditional grids in the design studios mentioned in Section § 4.1 , page 60 – see also (Beirão and Duarte, 2009) and Figure 3, page 62. However interesting the research on bottom-up generation may be, it was considered a specific line of

[17] *RectangleDissection* was used in the prototype implementation of Model B and three conference papers have already been written on this subject (Beirão, Nourian, and Mashhoodi, 2011), (Beirão, Nourian, and van Walderveen, 2011) and (Celani et al. 2011).

investigation within the research triggered by CityMaker and UIP formalism. This line of investigation should be treated as autonomous (although related) research. Two main aspects need further investigation for validation and design exploration, namely:

1 Bottom-up UIPs need to be further tested and explored, especially in a practical context. For instance, there are complex issues relating to recognition of neighbourhood structures which need specific research.
2 Exploring the use of bottom-up UIPs as regulatory systems for designing / managing flexible urban plan implementations.

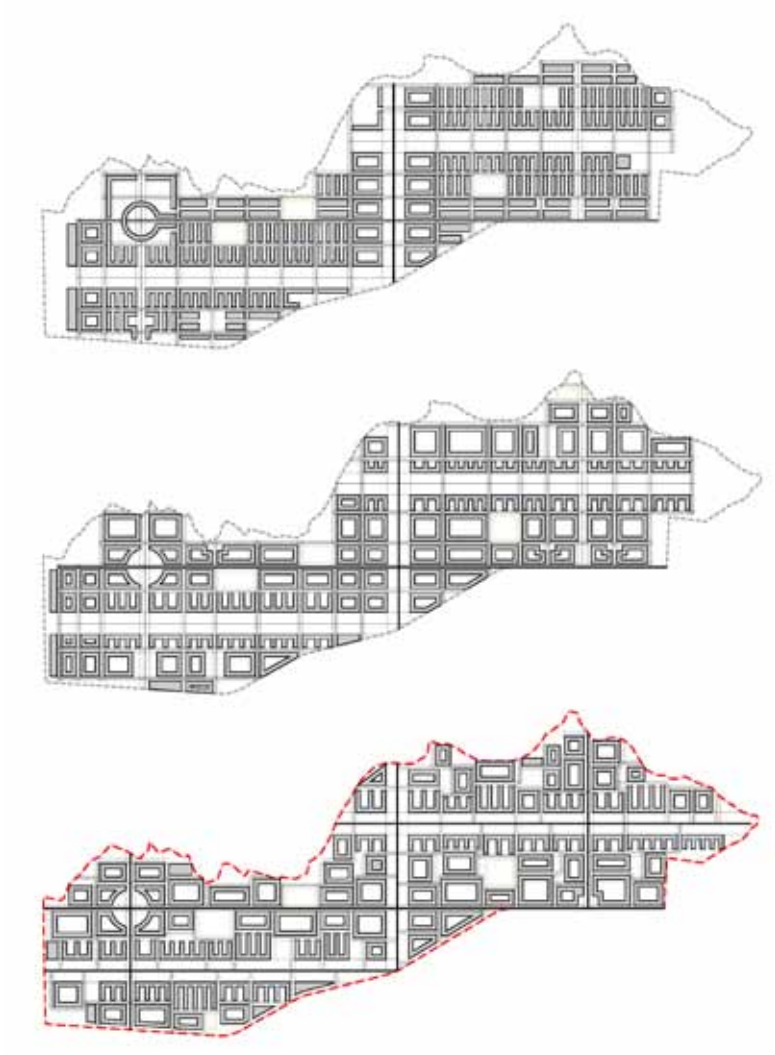


Figure 12
Design variations on Plan 1: 1- original plan; 2- varying h and w parameters with AddingAxes; 3- varying h and w parameters with AddingBlockCells.

§ 6.2.3 Applying UIPs to generate designs

Exploring design possibilities in CItyMaker is based on freely sketching many different ways of applying design moves to selected references within a given context. Both the references and the design moves may vary, as may the parameters or variables in each design move. Although the designer pursues a specific goal there are many possible and even optimal solutions, and the design exploration simply needs to find a way towards a solution space. Bearing this in mind, the aim is not necessarily to demonstrate the tool's capacity to find optimal solutions, although this may be achieved later when connecting formulation, generation and evaluation procedures, but to demonstrate the versatility of the design tool in exploring different design formalisations. In this sense, it is more important to show that, apart from being able to reproduce Chuva Gomes' plan, the UIPs also enable many different solutions to be designed. In particular, it has to be demonstrated that different results can be obtained by (1) selecting different R_{ef} references (2) applying different sequences of UIPs and (3) assigning different values to the parameters. These three optional manipulations provide the necessary space for design exploration allowing for: (1) designing the case studies; (2) designing plans with similar design languages for different places; and (3) designing different plans by developing other design languages, i.e. using the system in a creative way.

Some automated analysis could be developed to find or suggest references in a design context. However, this first step has been considered a manual one[18] . The designer selects elements of the available representations to define as references for the design. References (R_{ef}) can be focal points (e.g. a hill top), lines or polylines (e.g. an existing street, a ridge, a water line) and polygons (e.g. a building). The concept of focal point is defined as a geometrical position with a tolerance corresponding to the tolerance used by designers when sketching ideas in pencil or using a freehand tool. This means that the rules are structured to accept a certain flexibility with regard to the geometric position of the focal point. A building selected as a referential element (e.g. a historical building) can be treated by the rules as a focal point corresponding to its geometrical centre or as a polygon in which the longer line is used either as an alignment or as the basis for establishing a perpendicular axis from its middle point. Different selections as

18 In the context of the City Induction project some of the planned analytical procedures to be performed by the formulation model will be able to identify referential elements that the generation can use for designing plans. This feature is already available in 4CityPlan – the formulation module. However, there will always be a need to accept manually marked references to accommodate typical designer idiosyncrasies. As such, marking references was always considered from the viewpoint of generation as a manual procedure. This characteristic embeds interpretative subjectivity into the system in a very direct way, promoting a high level of flexibility and subjectivity in the generation of designs. These are the kind of decisions that should be left open to design experts.

well as different interpretations of the options will obviously produce different results. The options are therefore provided for the designer for disambiguation.

To obtain Plan 1 (Figure 7) a particular sequence of UIPs is applied to the initial conditions (I_s and R_{ef} s). In addition, the parameters to be applied should follow some of Chuva Gomes' design characteristics; in Plan 1, for instance, all blocks are the same size (50m x 80m). The sequence of urban induction patterns applied here is: *Cardus* (x 1) => *Decumanus* (x 1) + *Promenade* (x 1) => *AddingAxes* (x 1) (applying the optional rule which creates a higher hierarchy axis each time the generated axis meets the region of an *Ref* point) (x 4) => *AddBlocktoCells* (adds islands) (x number of cells) => *AdjustingBlockCells* (x number of cells crossing the boundary) (x 1) => *AddPlaza* (x 1) => *GeneratePlaza* (x 1) => *InsertPublicSpace*, option: *Sq2* (x 3) => *InsertPublicSpace*, option: *Sq3* (x 4) => *InsertPublicSpace*, option: *Sq4* (x 6) => InsertPublicSpace, option: *Sq5* (x 3) => *Square*, option: *SquarefromBlockSubtraction* (x 3) => *Square*, option: *SquarefromBlockTrim* (x 4) => *Square*, option: *SquarefromCornerTrim* (x 6) => *Square*, option: *CutPublicSpaceinBlock* (x 3) => *BuildingHeadingAxis* (x 5) . The integers in parentheses indicate how many times the UIP is applied to obtain Plan 1. Figure 13 shows the derivation of Plan 1 following this sequence of UIPs. The reader should consult Appendix 2 to follow the rules. The derivation in Figure 13 was simplified to the main intermediate steps.

MainAxis is an initial UIP containing three options as referred to in Table 6 – *MainAxisistheLongerLine*, *MIAxis* and *Cardus*. The initial UIPs are those able to recognise the available initial shapes, which are the R_{ef} elements and the intervention site limit I_s. The algorithm for the 3 options is basically the same and only the last step changes. They take all the selected R_{ef} elements and draw all the possible axes based on these elements. These axes are trimmed outside I_s. The longer axis defines the length l_{ax}, the longer axial length. Only the longer axes from the complete set of axes will be used in the next generation steps. The criterion for selecting the set of longer axes can be manipulated by the designer by changing the relative length of axes in comparison with the longer axis. From this set the designer is left with the 3 options: *MainAxisistheLongerLine*, *Cardus* and *MIAxis* (Most Important Axis).

MainAxisistheLongerLine selects the longest available axis from the set of proposed axes. *Cardus* selects the one closest to a north-south direction. *MIAxis* selects the most important axis[19] . References, R_{ef} labels, used by the selected axes are erased so that a coinciding axis may not be generated. *MainAxisistheLongerLine* and *Cardus* can be applied only once.

19 The use of weighted references has been considered as another possible way of selecting the axis instead of a random decision and it is closer to the real reasoning of a designer. The set of weights *W* is already considered for this purpose in the formal definition of a UIP. This principle is already being implemented in the software prototype.

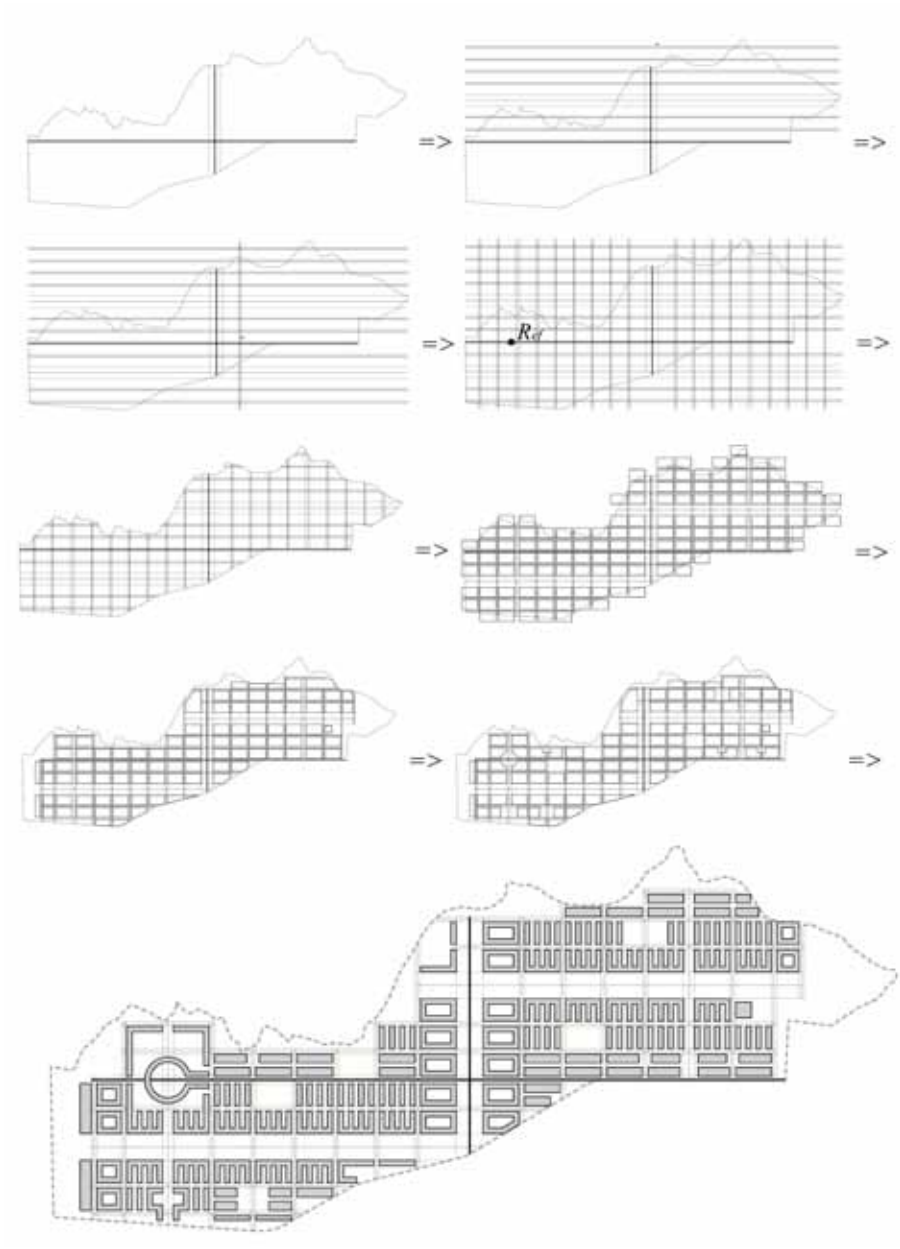


Figure 13
Derivation of the plan for Praia using AddingAxes. The steps indicated here compress the application of several rules, as described in the text.

OrthogonalAxis and *Decumanus* correspond to a second stage in UIP applications. They are applicable only if there is a main axis a_1 available or a *cardus*. *Decumanus* applies only if a *cardus* has been generated and only once in the whole design. *OrthogonalAxis* can be applied several times until there are no more references.
CompositionalAxis can be applied as long as there are still references to be used by generating a_2 axes following a similar algorithm as *MIAxis* (see rules in Appendix 2). *CompositionalAxis* can be used to generate non-orthogonal compositions.
The first level representations generated by these UIPs are axial representations of streets belonging to the **AN** (Axial Network) object class in the ontology and they basically represent 4 types of axes a_1, a_2, a_3 and a_4 corresponding to four distinct hierarchies. Other classifications can be added to the streets, detailing the street characteristics throughout the generation by adding attributes that change their configuration. In Plan 1 the architect decides to define the *decumanus* as a promenade and this feature is applied to all the a_2 axes in the plan, i.e. 3 times.
The UIPs *AddingBlockCells* and *AddingAxes* can be applied as soon as there are two orthogonal axes in the design. The parameters *h* and *w* correspond to the length and width of the urban block respectively. These parameters can be set as a fixed value, as Chuva Gomes does in Praia ($h = 80m$ and $w = 50m$), but can also be set as an admissible range (for instance: $60m \leq h \leq 120m$ and $40m \leq w \leq 100m$). In this case the results of applying these UIPs are different and their purpose becomes different in terms of design intentions. Although all the situations initially explored applied *AddingAxes* only to orthogonal axes, the rules can actually be applied to any two intersecting axes, generating regular grids following constant but non-orthogonal angles at junctions.
The derivations shown in the next section as well as the UIP grammar rules have already been published in some conference and journal papers and the subsection was adapted from these papers. The generation of Plan 1 generating the grid by *AddingAxes* was published in (Beirão, Duarte, and Stouffs, 2009). A complete generation of the same plan showing all the applied UIPs until the end of the grid generation is shown in this paper, this time generating the grid by *AddingBlockCells*. (Beirão, Duarte, and Stouffs, 2011) shows variations on the same plans obtained only by applying variable parameters to the block length and block width instead of the 50m x 80m used in the Praia plan. This variation in parameters proves that even simple parameter variations allow for wide exploration of design spaces. Changing the UIP sequence and other parameters extends the exploratory space available even more, allowing designers to explore the design system in terms of their own design language, needs or design concerns (Janssen, 2006). Some aspects of this work are shown in this thesis, together with the application of some other UIPs, completing the overview of the possibilities of the design system. The UIPs are shown in Appendix 2. The main idea is to demonstrate that the design system as it is conceived is able to generate the plans used as case studies and variations on these plans for different contexts, supports an interactive reflective design practice and allows designers to explore their own design concerns or individual interests. However, it should be pointed out that this design space is finite

and bounded by the sum of all the possible combinations of UIPs available in the system's pattern library (this library is reproduced in Appendix 2). Nevertheless, the system does not need to be infinite, just customisable so that designers feel that the tool allows them freedom to explore new design spaces. Some hints on how to achieve this are provided in this chapter in Section § 6.3 .

§ 6.2.4 Exploring variations in designs

At the beginning of the generation process, the designer is prompted to define a minimum set of values that are used by the generation module as input values for specific parameters in the Urban Induction Patterns. In terms of the generation module, these parameters can be set directly by the designer, although the formulation module is supposed to complete a table of specifications with such parameters as input data for the generation. In the next generation steps, in particular the exploration of grids, a few other parameters must be defined for the rules to use in *AddingAxes* and *AddingBlockCells*. These parameters are the block length *h* and width *w*, and the street width defined for the hierarchy of compositional axes, a_1, a_2, a_3 and a_4. The latter widths can be altered during generation if an axis is transformed into a specific street type, for instance a *Promenade*, such as the three promenades found in Plan 1. However, the street widths are set as fixed values, whilst *h* and *w* can be set as a range of values. It is this permitted variability that makes the grid generator UIPs so interesting to explore.
AddingAxes will be examined first, as this is an easier example. Beirão et al (2009c) presents the rules and the derivation of Plan 1. The rules are reproduced in Appendix 2. The derivation shown in Figure 13 applies the same values for block length and block width. Figure 14 shows a different derivation by applying an admissible range for the parameters *h* and *w*, such as the ones suggested above, in which variations appear in the grid whilst the typical orthogonal grid appearance and street continuity is maintained. The derivation in Figure 14 shows one possible solution resulting from the use of different values assigned to h and w randomly chosen from the stated range of values. No other function constrains the rule application in this example. However, a specific function could be used to set specific criteria for defining the distances between axes. For instance, some plans have larger blocks in the centre and smaller ones in the periphery, as is the case with the IJburg plan[20] . At the end of the generation sequence, squares are applied following different algorithms for the generation of public space. The last UIPs apply two different building typologies to

[20] This feature was implemented in Model B shown in Chapter 7.

blocks. These rules are not explained but their application is shown because it improves the legibility of the resulting urban plan. Apart from the fact that the rules were allowed to randomly assign the h and w parameters, all the other steps in the design attempt to produce a fair replication of the Praia plan, so that the result underlines the difference between using fixed or variable parameters to define block sizes. The derivation in Figure 14 was simplified to the essential steps.

AddingAxes is a sub-grammar Γ' of the generic urban grammar Γ being used. Γ' is composed of two parallel grammars, Γ_1 and Γ_0 – $\{\Gamma_0,\Gamma_1\}$ – in which S_1, L_1 are the shape and label sets in Γ_1 respectively and both are objects of the **AN** class in the ontology. S_0, L_0 are the shape and label sets in Γ_0 respectively and they are both objects from set E_0. E_0 is the set of existing elements in the territory.

The first step in the derivation already demonstrates the results of applying *Cardus*, *Decumanus* and a *Promenade* and shows all the R_{ef} points still to be erased. These R_{ef} points will be used to attribute a higher level of hierarchy to some of the axes generated. The second step starts the application of *AddingAxes*. Steps 4-5, 9-10, 15-16 and 18-19 are the steps in which the R_{ef} points change the hierarchy of the axis to a higher level. Steps 4-5 and 9-10 apply a new *Promenade* to the axes passing through the first two of these referential points. Due to space restrictions, some of the repetitive steps were condensed. The last steps apply the generic blocks (UIP - *AddBlocktoCells*, step 20), adjust the blocks to the site boundaries (UIP - *AdjustingBlockCells*, step 21), create squares by subtracting some of the blocks (step 22), create the main plaza (step 23), create smaller squares by shrinking the block and reducing one of its parameters (step 24) or by subtracting some corners at a junction (step 25), and, finally, replace the generic blocks with two different types of building occupation, the closed block, composed of buildings surrounding the entire block, and a spine-like building occupation with the continuous side facing the main streets (step 26).

Appendix 2 shows the UIP *AddingAxes* (UIP 009) with Rules 3a, 4a, 4c, 5a, 6a, 6c and 8. Some rules are omitted because they are symmetrical to others, namely Rules 3b, 4b, 4d, 5b, 6b and 6d, which are symmetrical to Rules 3a, 4a, 4c, 5a, 6a, and 6c respectively. *AddBlocktoCells* and *AdjustingBlockCells* are also reproduced in Appendix 2 – UIPs 024 and 025.

Rule 1 in the *AddingAxes* UIP maps a temporary coordinate system, *x*0*y*, into two perpendicular axes a_n and a_n'. In this case, the axes are the ones generated by the patterns *Cardus* and *Decumanus*. Rule 2 extracts the maximum and minimum coordinate values from I_s taking the new coordinate system into account. The points are: $\max_x$ = maximum *x* value of I_s in *x*0*y*; $\max_y$ = maximum *y* value of I_s in *x*0*y*; $\min_x$ = minimum *x* value of I_s in *x*0*y*; and $\min_y$ = minimum *y* value of I_s in *x*0*y*. These values will be used to frame the generation within the space defined by these coordinates.

Rule 3a adds a street axis - a_4 - parallel to a1 or to the cardus. The labels ▲ and ▼ are used to define the recursive application of Rule 4 (a, b, c and/or d) and indicate the direction in which to apply the next rule. Rule 3b is symmetrical to Rule 3a and is applied in the negative y coordinate direction. Rule 4a adds a street axis - a_4 - parallel to an a4 axis labelled with ▲, erases the label on the original a_4 axis and creates a new

▲ label on the new a_4 axis. The rule applies recursively until it falls outside the intervention site, i.e. while $y<max_y$ where y is the new a_4 *y* coordinate referred to *x*0*y*. Rule 4b is symmetrical to Rule 4a and is applied recursively in a similar fashion. If a selected R_{ef} point or element is within the region of a new axis, the designer is prompted to decide whether he wants to apply a new level of hierarchy to this new axis. This application is optional but if the designer does choose to use it, the axis is transformed into an axis of a higher level of hierarchy. Rule 4c applies this procedure and Rule 4d is symmetrical to it. Steps 2-11 in Figure 14 show the application of these rules.

Similar rules apply to the orthogonal axis a_2, or the *decumanus* to generate an array of perpendicular axes along the *x* coordinate. Rules 5a, 5b, 6a, 6b, 6c and 6d are used to generate these axes. Steps 12-19 in the derivation show the application of these rules. The rules shown here are exactly the same as those used to generate the Praia plan, except that the values given to the parameters *h* and *w* are different in each iteration. The parameter values were defined randomly merely to explore design variations. However, this input could be informed through the formulation module, with specific values taking contextual data extracted from the site into account.

Rules 7a, 7b, 7c and 7d erase the ▲, ▼, ► and ◄ labels, respectively if they fall outside the framed area. Rule 8 trims the axes outside the I_s limit and Rule 9 returns to the original coordinate system.

AddingAxes works with variable parameters in more or less the same way as it does with fixed values. On the other hand, *AddingBlockCells* behaves in an entirely different way. Defining each cell in the generation with different parameters creates a huge range of possible variations and a lot of unpredictability throughout the different steps of the derivation. In order to deal with this complexity, the set of rules in this UIP had to be expanded in comparison with the ones previously shown in Beirão et al (2009), in which the goal of the grammar was the replication of the Praia plan. *AddingBlockCells* shows a series of adjusting rules to solve all the maladjustments resulting from parameter variation in the application of the main rules.

Again assuming that the generation will use an admissible interval for setting the values for the length *h* and width *w* of the block (again: $60m \leq h \leq 120m$ and $40m \leq w \leq 100m$), the block cell is defined by the block parameters plus the streets confining the block, which may be all different in some extreme cases. All the situations can be seen in the adjustment rules shown in pattern 010 in Appendix 2. Figure 16, showing the derivation, helps illustrate these situations. Since each iteration can have different *h* and *w* values, the configurations of the design may contain many different variations, making recognition of the left-hand side of the rules extremely difficult to manage. Liew (2003) provides seven descriptors to be used in the rule application process to solve some problems regarding conditional matching in rule application. Specifically, with regard to contextual requirements he proposes the use of a descriptor 'zone' which associates an area in a schema with a predicate function. He gives the example of a *void* function which states that a certain area must be empty of all shapes for the rule to apply or specifies a conditional application when certain occurrences are detected in the 'zone' area. Because of the unpredictability of the

AddingBlockCells grammar a similar descriptor was used in its rules. The rule checks the context locally every time it is applied. The main rules for *AddingBlockCells* are basically the same six main rules in Beirão et al (2009) but the descriptor zone was added to three of them creating a set of adjustment rules which consider all possible occurrences during the generation.

Rule 1 (see pattern 010 in Appendix 2) places 4 labels ● at the intersection of an a_1 and an a_2 axis or the intersection of a *cardus* and a *decumanus*. Rule 2 starts the cell derivation, erasing one of the labels ● and adding two orthogonal a_4 axes. The cell width *v* and cell length *u* are defined by the values randomly chosen for *w* and *h* respectively, added to half the width of the streets that flank them. Rule 3 is applied recursively until all labels ● are erased. The values for *w* and *h* are randomly chosen from the admissible range in each iteration. To ensure recursive behaviour, Rule 3 erases the original label ● and places another label ● next to the new a_4 axis on the right-hand side of the cell so that it can be used by the same rule in the following iteration. The rule creates a second label ● in the top left corner of the cell above the new a_4 axis, to be used later by Rule 5. Labels ● are only recognised by Rule 5 and their adjustment variations. To summarise, Rule 3 creates cells along the a_1 axis or any a_n axis parallel to the *x* coordinate where $n \in \{1,2,3\}$, until a vertical axis a_n' is found in the area checked by the *void* zone. The rule applies when the *void* predicate is true. There are 4 different situations that can occur if the *void* predicate is false. These 4 situations are the adjustment rules 3A_1, 3A_2, 3A_3 and 3A_4. Rules 3A_1 and 3A_2 adapt the size of the new cell or the previous cell to meet the a_n' axis and create a new ● label on the right-hand side of a_n' to allow a new generation sequence to start. Conversely, Rules 3A_3 and 3A_4 move the a_n' axis until it fits the length *u* of the cell. These rules can be applied only if an is the main axis in the design or, in other words, if a_n was generated by *Cardus* or *MainAxisistheLongerLine*. This guarantees that an a_n' axis will not be moved more than once. Once again, a new label ● is placed on the right-hand side of a_n'.

Type A rules are all the adjustment rules that detect the presence of axes (objects from the **AN** object class) inside the *void* zone. Other types of adjustment rules react, for instance, to existing constructions, elements of set E_0, which are either streets or buildings. However, these rules are not shown, as there are no existing buildings within the Praia site. The important point to make clear at this moment is to stress that the 'zone' predicate can be used to add conditional behaviour to the design rules, developing automated reactions to different kinds of occurrences within the zone area. Thus, this mechanism can be used to add selective rules to respond to the existence of buildings, rivers, paths, trees or bushes, existing obstacles or even topographical deformations. The grammar of this pattern can be therefore extended to react to such occurrences. However, this requirement shows the importance of having customisable processes to extend rule application. Formalisms to develop customisable UIPs will be further discussed in the explanation of *DefineUUnit* and in the discussion chapter.

Like the 3A rules, Rules 4A_1 and 4A_2 adapt the width *v* of the cell to meet the detected axis a_n' parallel to the main axis or the *cardus*. A label ● is created at the top of

a_n' to allow another generation sequence to start in another area of the plan (see derivation in Figure 16. If a_n is a *decumanus* or the first orthogonal axis applied in the derivation, Rules 4A_3 and 4A_4 can be applied alternatively to adjust the position of a_n' to the cell size *v*. Like Rules 3A_3 and 3A_4, Rules 4A_3 and 4A_4 can be applied only once per axis.
While applying Rule 5 several occurrences may be detected inside the *void* zone, namely:

- Rule 5A_1 and Rule 5A_4 detect the presence of one a_4 axis.
- Rule 5A_2 and Rule 5A_3 detect the presence of one a_n axis.
- Rule 5A_5 detects the presence of two a_n axes.
- Rule 5A_6 and 5A_9 detect the presence of one a_4 axis and one a_n axis.
- Rule 5A_7 detects the presence of one a_4 axis and two a_n axes.
- Rule 5A_8 detects the presence of two a_4 axes and one a_n axis.

Rule 5A_1 detects the Δx length of penetration of an a_4 axis inside the *void* zone and, depending on the value of Δx, produces two separate results. If $\Delta x \leq u/2$, Rule 5A_1a generates a new cell creating two axes reducing the cell length *u* to *u'* so that $u' = u - \Delta x$. This rule creates a new ● label in the top left-hand corner of the cell. If $\Delta x > u/2$, Rule 5A_1b simply erases the existing ● label and creates a new one above it at a *v* distance in order to allow continuity of cell generation.
Rule 5A_2 reduces the cell size *u* to *u'* so that $u' = u - \Delta x + d_2 + d_4$ creates an a_4 axis closing the cell and placing a new label ● in the top left-hand corner. The cell dimensions become $u' \times v$. Rule 5A_3 produces a similar result in the *y* coordinate direction, creating a cell with dimensions $u \times v'$ in which $v' = v - \Delta y + d_2 + d_3$. Note that this rule simply erases label ● and does not create a new one.
Rule 5A_4 is similar to 5A_1 but it generates the reduced cell adjacent to the right side of the *void* zone instead of the left side. In fact, it is using the empty space left in the *void* zone. *u* is reduced to *u'* through the relationship $u' = u - \Delta x$.
Rules 5A_5 and 5A_7 simply erase the label ●. They are termination rules. Rules 5A_6 and 5A_8 generate a new cell, shortening both parameters *u* and *v* to *u'* and *v'* respectively. They both create an a_4 axis and do not create new ● labels. Rule 5A_9 reduces the cell length u to u' following the equation $u' = u - \Delta x_1 - \Delta x_2 + d_2 + d_4$, producing a cell with dimensions $u' \times v$.
The derivation in Figure 16 shows the generation of an urban plan using *AddingBlockCells* as the algorithm for grid generation. The derivation starts in Step 1, already demonstrating the result of applying *Cardus* + *Decumanus* + 4 × *OrthogonalAxis* + 3 × *Promenade*. Step 2 applies Rule 1 by placing 4 ● labels. Step 3 applies Rule 2, generating the first cell, erasing one of the ● labels and creating two more, associated with each of the newly created a_4 axes. Steps 4-6 apply Rule 3. Rule 3A_3 is applied in step 7. Note that a new ● label is created on the right-hand side of the a_n' axis. Step 8 applies Rule 4 and Step 9 applies Rule 4A_1, adapting the cell size *v* to the available circumstances. Steps 10 to 17 apply Rules 5 and 5A until all the ● labels are erased. Generation is then terminated in this section of the plan. The other

sections are generated in a similar fashion using the available ● labels, starting with Rule 2 and ending with the exhaustion of labels ● and ●. Note that every label falling outside I_s is erased. Step 23 is the last cell creation step.
The complete set of rules allows the urban grid to be generated without conflicts but the final results still need amendments, namely by aligning or connecting a few streets and placing blocks within cells. However, these amendments are produced by other UIPs. The derivation clearly shows some of these problems and how they are solved in Steps 23 to 26.
In Step 24 the UIP *AlignStreets* is applied. The result of the generation using *AddingBlockCells* produces several situations in which a_4 axes connect to a_n axes that are very close to each other but not actually aligned. These situations are a little unsound as they create unusual and probably unwanted junctions. An urban designer will probably want to have either a clear 'X' junction or a clear 'T' junction. This problem can be solved in two ways. One involves defining a minimum module for the varying parameters *h* and *w*: if the minimum variation is 10 meters, for instance, this will restrict the distance between street connections to 10 meters or multiples of 10 meters. The other solution is to apply the *AlignStreets* pattern. *AlignStreets* takes two axes connecting a higher level of hierarchy axis at points closer than three halves of their width and moves one of the axes to align with the other, creating a 'X' junction instead of a 'T' junction. The choice of whether to move one or other of the a_4 axes depends only on the degree of connectivity of the axes. The axis with the least connectivity is chosen as the one to be moved. The criteria for evaluating the degree of connectivity are as follows: (1) a_4 connects with another a_4 street, (2) a_4 has a corner connection with another a_4 and (3) a_4 aligns with another a_4 segment (see pattern 013 in Appendix 2). The degree of connectivity increases from (1) to (3).
In steps 25-26 all the axes falling outside I_s are either trimmed (step 25) or erased (step 26). Very small bits of a_4 axes are left from step 25. All those smaller than half of the lower value defined for *h* are erased – step 27. In this step it can be seen that some of the streets are still not connected and do not contribute to the consistency of the street network (see detail in Figure 17). *ConnectStreets* is used to correct some of these inconsistencies in the grid generation. Step 28 shows the result of applying *ConnectStreets* (see rules in Appendix 2 – UIP 015). In step 29 the cells are filled with abstract blocks using *AddBlocktoCells* (UIP 024 – Appendix 2). Step 30 adjusts or erases the blocks on the borders of the plan applying the pattern *AdjustingBlockCells* – UIP 025 in Appendix 2. Every block falling outside the I_s area is erased. Step 31 shows the final result of applying several UIPs generating squares or public spaces.
The generation model considers 6 types of squares which can be found in urban plans. This classification of squares is related to the way in which they are generated rather than the identification of a particular morphological type.

The 6 types are:

Sq1 A Plaza or a Main Square is a planned, closed and intentionally designed public space with typical applications in classical urban design. The round plazas found in Chuva Gomes's plans are an example of this type. It is a structural element in urban design and may be applied before the grid is generated. Two UIPs are used for generating plazas – *AddPlaza* and *GeneratePlaza* (UIPs 020 and 021 respectively in Appendix 2).

Sq2 Corresponds to the subtraction of a block in a grid (e.g. as in Plan 1). Uses UIP 022 *InsertPublicSpace* (option *Sq2*) and UIP 023 *Square* (option *SquarefromBlockSubtraction*).

Sq3 A square resulting from the subtraction of part of a block in a grid (e.g. the public space in front of the Seagram Building in New York). Uses UIP 022 *InsertPublicSpace* (option *Sq3*) and UIP 023 Square (option *SquarefromBlockTrim*).

Sq4 A square produced by subtracting shapes from the corners of city blocks in a junction (e.g.: the Barcelona Cerdá plan squares). Uses UIP 022 *InsertPublicSpace* (option *Sq4*) and UIP 023 *Square* (option *SquarefromCornerTrim*).

Sq5 A square that results from opening an inner courtyard in a block (e.g.: the Spanish Seville patios). Uses UIP 022 *InsertPublicSpace* (option *Sq5*) and UIP 023 Square (option *CutPublicSpaceinBlock*).

Sq6 A public space formed out of a remaining open space in an irregular grid (e.g. the typical mediaeval square).

Sq6 is not generated. It is a free space remaining from the intersections of a street crossing a grid. The recognition of such spaces involves some intelligence on the part of the system but it can be solved by filtering parts of blocks resulting from intersections or simply filtering the smaller blocks in an irregular grid generation. The main criterion is that *Sq6* squares are usually small public spaces.

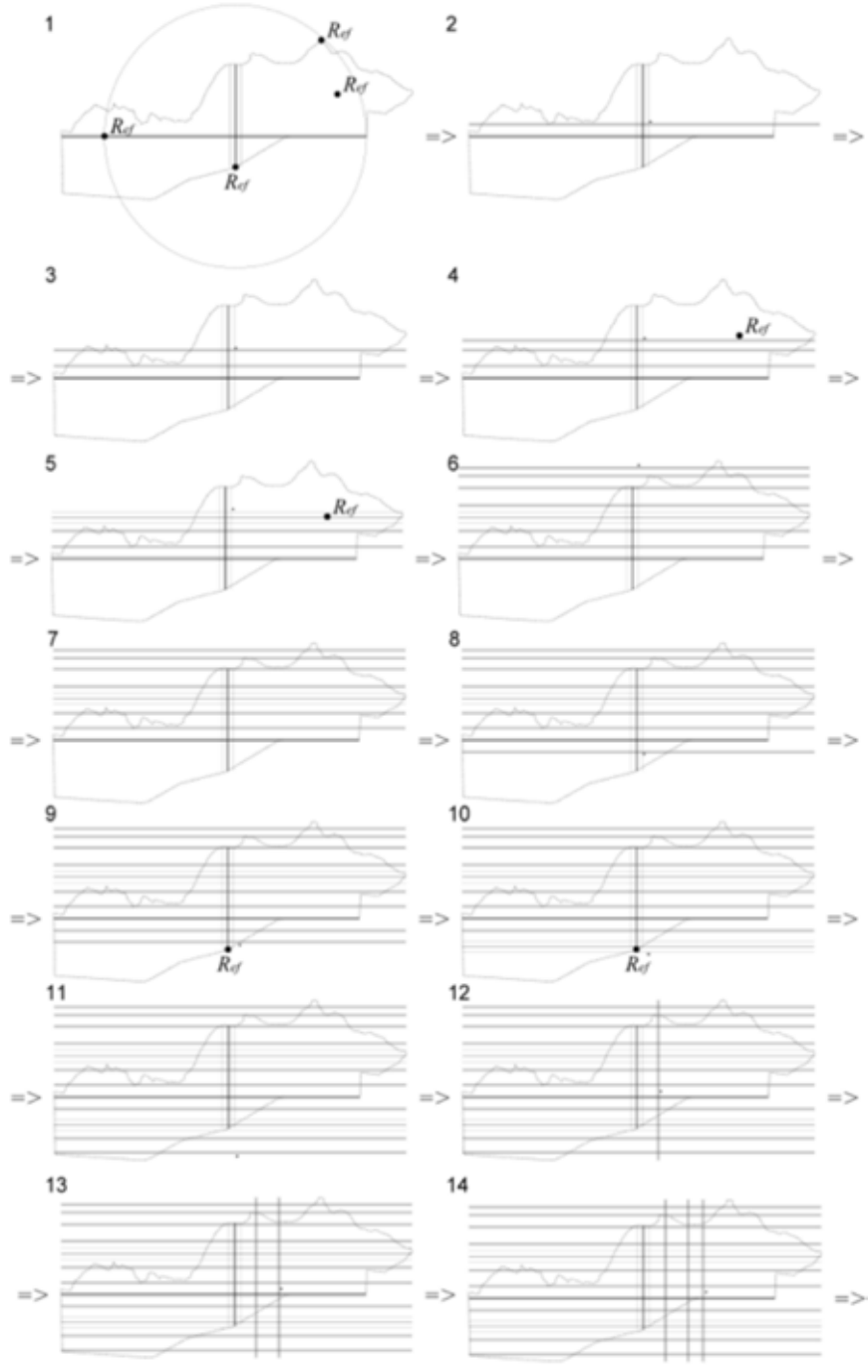
The public space patterns work in two separate steps corresponding to two distinct procedures: the first step introduces a label for locating a square type (*AddPlaza* or *InsertPublicSpace*) – the location step, whereas the second step generates the square type (*GeneratePlaza* or *Square*) – the generation step. The location step is essentially a programming step which is fundamental to the generation of squares. Locating a square implies a thorough analysis of the design context and the identification of contextual needs before a decision on location is made. CItyMaker, as an urban generation system, keeps the location task manual for the designer to decide, based on the principle that the designer will perform the necessary analytical tasks before locating the squares. However, combined work on the location of squares has already been developed involving the collaboration of Montenegro, the researcher in charge of the City Induction formulation model. In fact, the programme formulation interface is based on the inference capabilities of ontologies. The ontology infers the programmatic needs by examining data on the existing context conditions, against the

needs resulting from an input of estimated population growth for the intervention site. The ontology contains rules relating specific types of public spaces to the populations they serve, taking the distance from public spaces to people's dwellings into consideration. The system searches for optimal locations but also allows the designer to set the location of a particular public space if s/he so desires. This information can be inserted into CItyMaker to support the location step. However, it should always be possible to change the location manually. Both CItyMaker and 4CityPlan (the name given to the formulation model) have been developed following this concept. For details on the ontology implementation and regarding the management of public spaces, see Montenegro et al (2011). The generation step follows the rules shown in UIPs 021 and 023 in Appendix 2. The division of public space generation into two steps has two advantages: (1) it separates programming from generation and (2) allows for checks on whether all the predicate conditions for generation are fulfilled. The second advantage controls the system by preventing generation from starting if the predicate conditions (labels and shapes) are not totally fulfilled and erasing the previously defined label.

Steps 32-33 replace the abstract block with a few different block types showing the different possibilities for building occupation within the block. The block types used in these rules are the same as the rules used by architect Chuva Gomes in Plan 1.

It is important to stress that it is not the intrinsic qualities or weaknesses of the final plan that are the goal in the generation of variations, but rather a demonstration of the versatility of the UIP *AddingBlockCells* in generating different plans. This is accomplished by showing the results of randomly changing the *h* and *w* parameters for each iteration. The application of these parameters could be informed through other means, such as correctly formulated programming tools, in order to generate solutions following specific criteria.

Finally, it should be pointed out that the different UIPs only use objects from specific classes in the ontology. Likewise, the elements generated also belong to specific classes in the ontology and constitute different layered representations in the drawing. In very general terms, all the axes belong to the **AN** object class, all the generic blocks belong to the **BL** object class and all ▲, ● and ● labels belong to the **AN** attribute class.


1
R_{ef}
R_{ef}
R_{ef}
R_{ef}
2
3
4
R_{ef}
5
R_{ef}
6
7
8
9
R_{ef}
10
R_{ef}
11
12
13
14

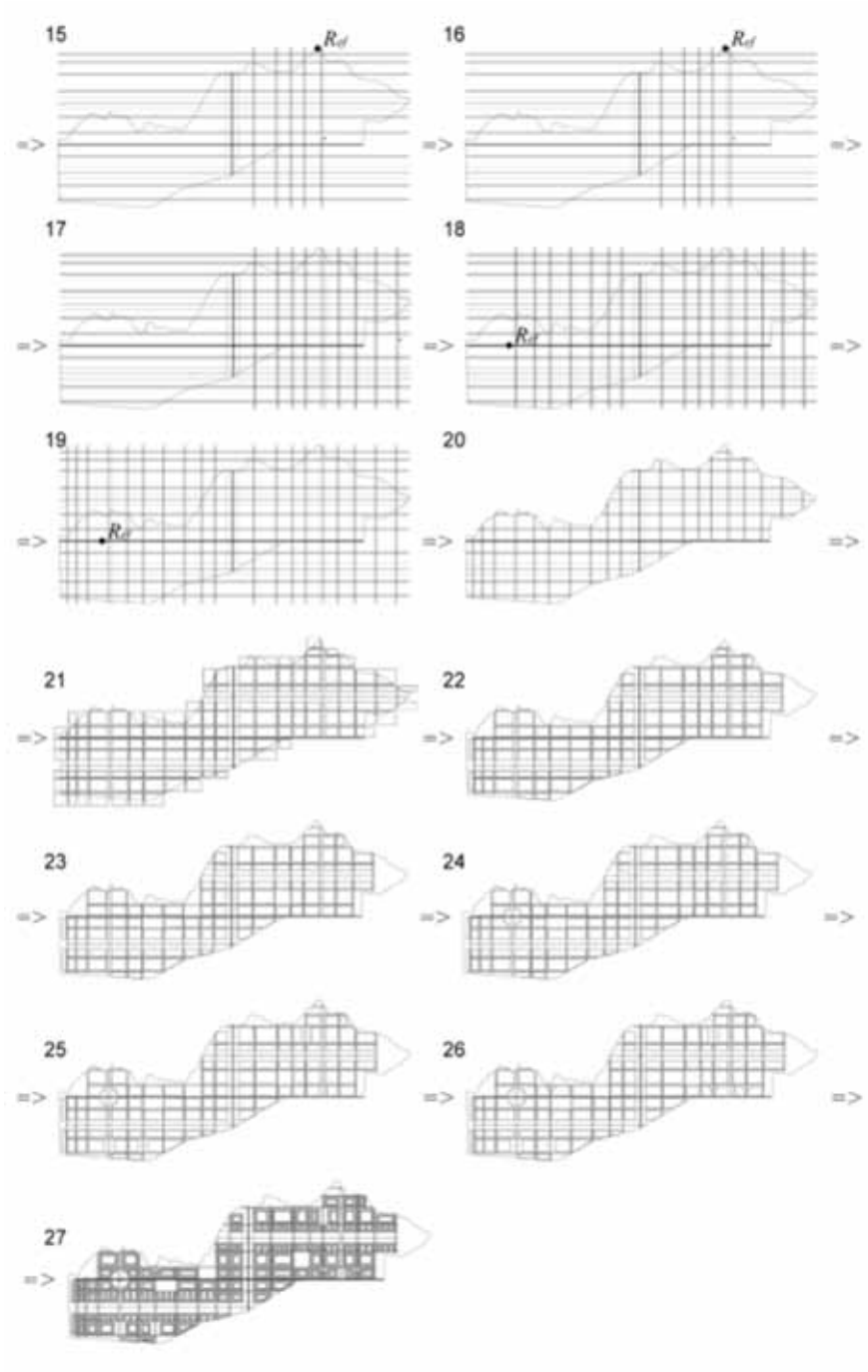


Figure 14
Derivation of the plan for Praia using AddingAxes. The derivation is simplified to the essential steps.

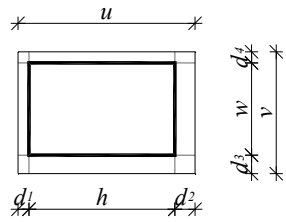
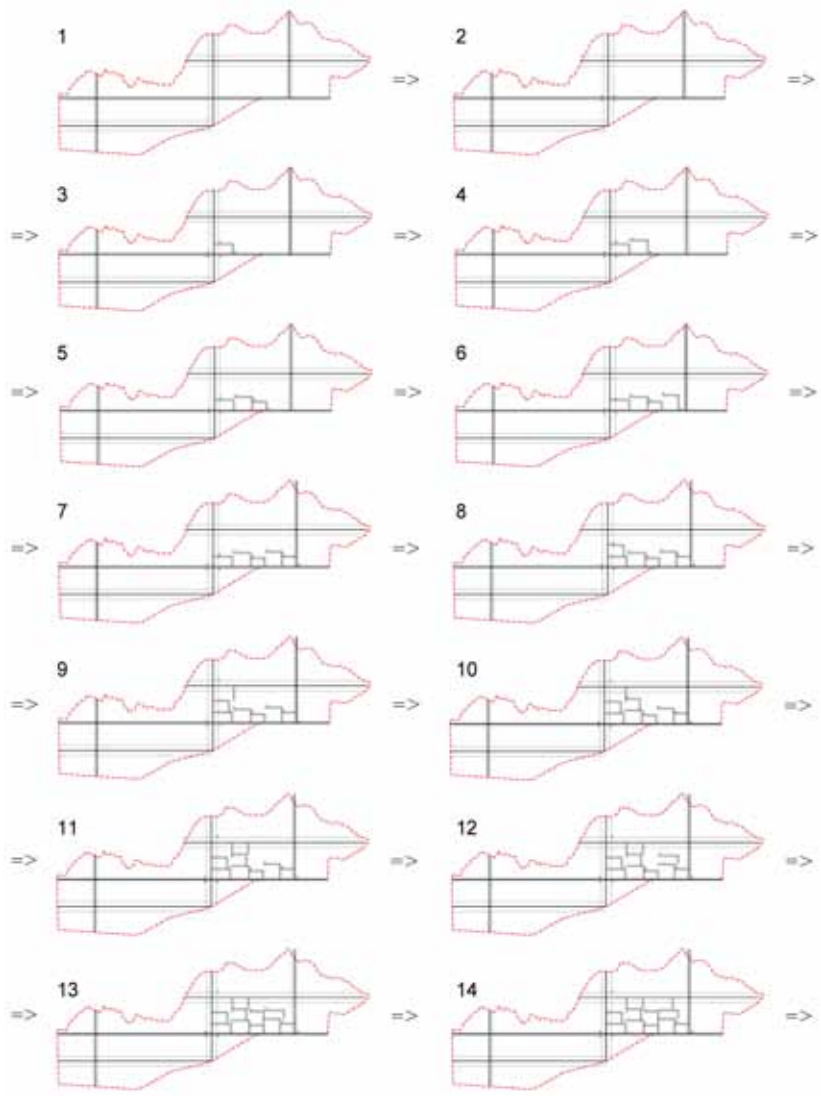


Figure 15
The block cell – parameters and labels



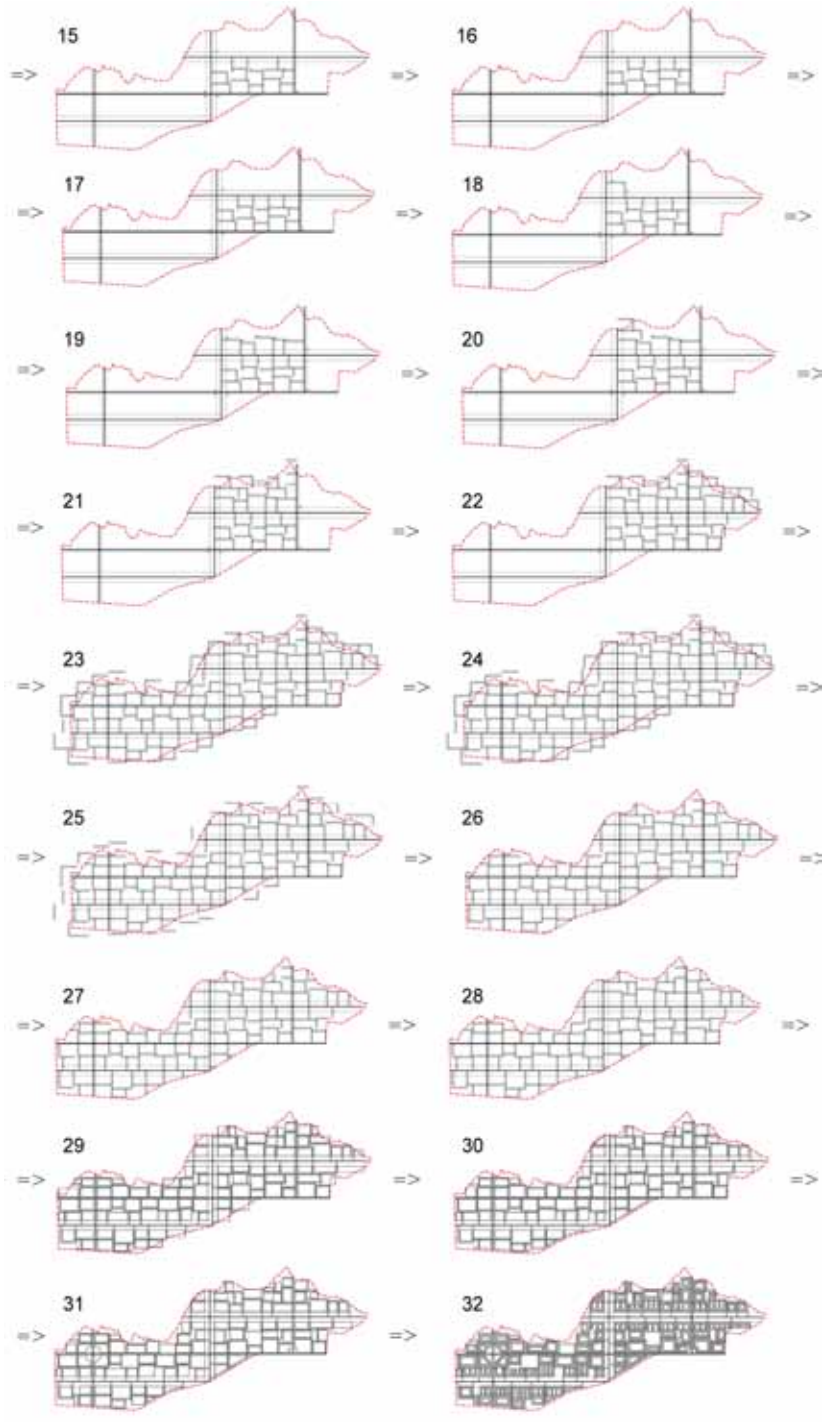


Figure 16
AddingBlockCells – derivation including adjustment rules

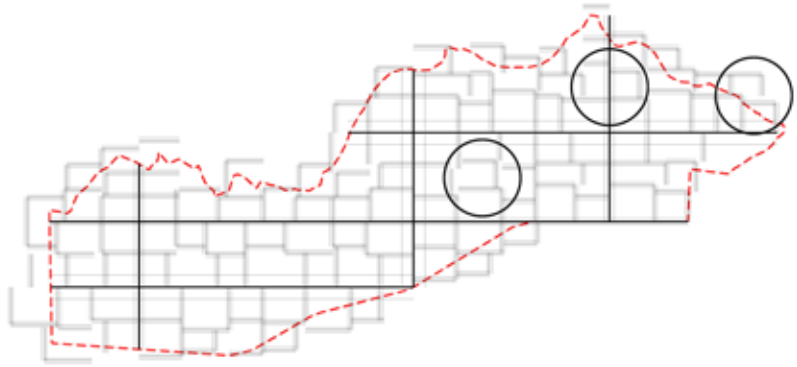


Figure 17
Detail of Step 23 showing some inconsistencies in the generation which are corrected by the UIPs that follow. The circles indicate different kinds of inconsistencies.

Design exploration with CItyMaker is guaranteed by a small set of characteristics defined in the system's structure. The design exploration options are available due to:

- differences in the context representations or changes in input geometries: different sites (I_s) / different referential elements (R_{ef}) / different classifications of existing elements / different weights attributed to elements;
- differences in the chosen combination of UIPs;
- differences in optional rules available in some UIPs (see Appendix 2);
- differences in parameter inputs.

In the derivations shown in this section, only the latter aspect – parameter input variation – was explored for producing design variation. However, all the other aspects can be used for this purpose. Varying all the options produces a great variety of solutions for a specific site. It also introduces an interactivity which allows the designer to express his/her interpretation of the design context, especially through the reference (R_{ef}) classification procedure.
Plan 2, for instance, shows a plan for a different location designed by Chuva Gomes, also the author of Plan 1. The design has a greater degree of complexity but the design moves used to generate it are more or less the same. The initial state of the design already reveals a higher level of complexity which to some extent explains the greater diversity of the plan. Figure 19 shows the guidelines and proportions for this design, as explained by the author. In Figure 18 it can be seen that at the beginning of the process the intervention site (I_s) is already subdivided into several areas due to the presence of a railway line and a main distribution road connecting two small towns located a short distance away from each other. It shows intensive subdivision of the intervention site into a set of urbanizable areas (U_a). There are also six housing blocks isolated from any other settlement, a small train station and an old farm house representing a

reasonably large private property. This set of conditions created a few isolated areas separated by noise protection areas in which only a few connections could be made. It is fairly easy to understand that Chuva Gomes applies very similar composition principles in each of these areas, starting with two orthogonal axes and generating rectangular grids based on the two axes, applying a grid by *AddingAxes*. The size of the blocks changes in each area according to different programmatic goals. One of the main axes in each area connects with another main axis in another area creating some urban continuity in spite of the significant barriers that both the railway and the main road create. The public spaces created in this plan are also quite similar to those found in Plan 1.

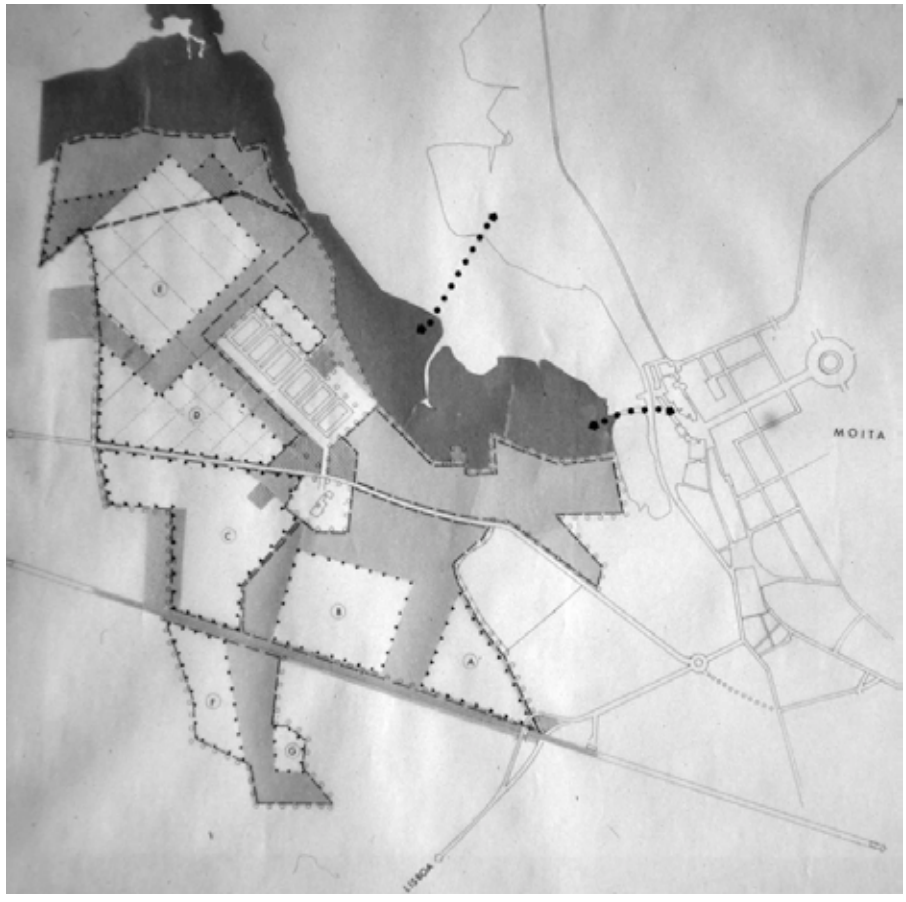


Figure 18
Sketch by Chuva Gomes showing the urbanizable areas for Plan 2.

The block shapes and sizes are obviously related to the housing types and programme proposed for each area. Housing types and urban morphology are related issues.

Although the relationship between type and morphology is not linear, Berghauser-Pont and Haupt present a very interesting study relating urban morphology and density measures. They compare a vast set of case studies by examining their density measures. This kind of study is able to accurately determine the sets of density measures that are related to certain morphological types. However, the study needs further development in order to relate building types to both density measures and urban morphology. If such relationships are correctly identified, at least establishing the average levels, valuable information may be obtained for automating some programmatic aspects of the generation. For instance, we may be able to develop either typology-oriented regulations for the grids generated or generate grids for specific typological goals.

Figure 19
Plan 2 – Qta da Fonte da Prata, Moita. Guidelines and proportions.

Block occupation / design can be rather complex to design as the IJburg plan blocks clearly shows. The next subsection focuses on urban induction patterns for generating block typology and shows the use of the discursive structure of UIPs.

§ 6.3 Urban Induction Patterns are discursive grammars – designing with semantics

Different types of urban blocks confer different appearances on urban spaces and are to some extent responsible for the type of social interactions that may emerge in specific formal environments. Most of the differences in character between the four case studies are due to the different block types used in each case. The two Chuva Gomes case studies (Plans 1 and 2 for Praia and Moita) are quite simple in terms of block types, whereas the two Dutch case studies show an extremely rich variety of block types. However, the IJburg plan (Plan 3) has a simpler approach to urban block typology than the Ypenburg plan, where blocks are complex compositions of buildings and public spaces. Apart from Ypenburg (Plan 4) all the other case studies have in common the fact that all the block types are transformations of a basic rectangular shape. This basic shape will be called the *island*. *Islands* are the empty spaces in the street network, or the negative of the street network. *Islands* are created in these spaces after grid generation by the UIP *AddBlocktoCells*. The *island* can later be replaced by a block type or subtracted (partially or totally) to create a square using the public space generation UIPs referred to in the previous subsection (see UIPs 020 to 023 in Appendix 2).
Plan 1 uses only the 4 block types shown in Figure 20. Plan 2 uses the same types but with a larger range in terms of size variation. Block size is related to variations in the housing programme, thereby showing a clear relationship between certain housing types and urban morphology. Smaller blocks are used, for instance, for densely occupied single family rows of houses. In Plan 1, all the blocks are $80m \times 50m$, whilst in Plan 2 there are extreme variations from large to very small sizes containing just one row of houses (see Figure 8). Figure 21 shows the parametric rules used in Plan 2 which apply the same block types as in Plan 1 but allow for variable parameter values instead of discrete values. In the IJburg plan (Plan 3), the blocks are of different sizes but they are all relatively large. This characteristic allows designers to explore different types of urban blocks, following different approaches for occupying the inner block spaces. Figure 22 reproduces some of the designers' pictograms to show the concepts underlying the composition of morphological block types.
The simplest case is the Praia plan – Plan 1. The generation of the grid was explained in the previous subsection; for the block generation, we will start with the output of the sequence of UIPs previously described. After the application of this sequence of UIPs, a point is reached in the generation where the plan assumes the appearance shown in

Figure 23a. Figure 23b compares this state with a drawing produced by the author of the plan as an explanation of his design procedure.
At this point in the design the general layout of the plan is set at the level of network definition, together with some additional features relating to the definition of public spaces. The *islands* only provide an abstract mass occupation without any other morphological definition. However, if a goal density had been defined for the plan, the average goal density per block could have been calculated or, if some differentiation is intended for the plan, this difference could have been defined at the beginning of the design process, defining the specific density regulation for different urbanizable areas (U_a – see Section § 6.1 , page 115). In any case, much of the character of the plan is defined through the design of the urban blocks.

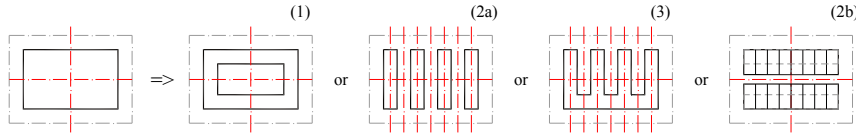


Figure 20
The 4 block types used in the Praia plan. The UIP AddBlockType replaces the island with one of the 4 types. (1) is a closed block, (2a) and (2b) are different versions of blocks of parallel buildings, the linear block, and (3) is a spine-shaped block.

The following procedure replaces an *island* with one of the block types, as shown in Figure 20. It shows the formal rules for Plan 1 in which an *island* is replaced by one block type. In Plan 1 the designer uses always the same parameters, a $80m \times 50m$ block. The shape rules in Figure 20 are particular assignments of values for the variables *h* and *w* of the rule schemata in Figure 21. Generically, it can be said that the main rule in every case is a simple replacement rule in which the occurrence of the *island* appears on the left-hand side and its replacement with a block-type schema on the right-hand side.

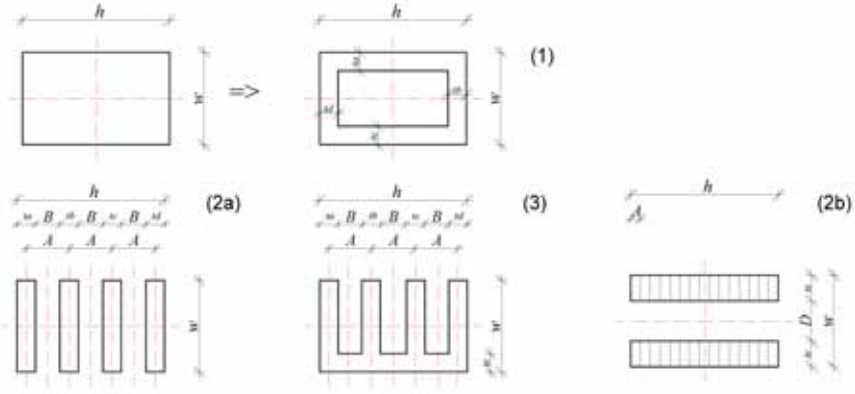


Figure 21
Rule schemata for the Moita plan block types – rectangular parametric blocks

The generation of a block type follows a three-step procedure: (1) *ClassifyUUnitCells*; (2) *DefineUUnit* – option: *BlockType*; and (3) *AddUUnitbyLabel* – option: *AddBlockType*.

1 A classification of block hierarchies in the grid which assigns different labels to the blocks depending on their position in the grid. The UIP applied here is called *ClassifyUUnitCells* and, as a result, it adds labels to the *islands* in order to guide the generation of block typology. Appendix 2 shows a simplified version of this UIP. The final product of applying *ClassifyUUnitCells* is a plan layout in which islands are added with a label identifying the block type that will replace the *island* (Figure 29). This label is basically an attribute which sets the priority for using a specific block type according to its particular location in the plan. The decision on the location of these priorities can be defined by the formulation module, or, in the absence of this input, directly by the designer. However, some alternative algorithms can be used to distribute the hierarchical values according to predefined criteria based on the characteristics of the elements already present in the design, for instance, placing specific block types next to promenades as seen in Plan 1 (Figure 7 and Figure 30). Blocks can also be distributed according to any other fixed criteria, for instance distributing density by applying higher densities close to promenades and lower densities in the peripheries. The distribution process can be based on assigning weights to specific components of the design and then distributing density throughout the blocks, increasing the density on the basis of the attraction of heavier locations[21].

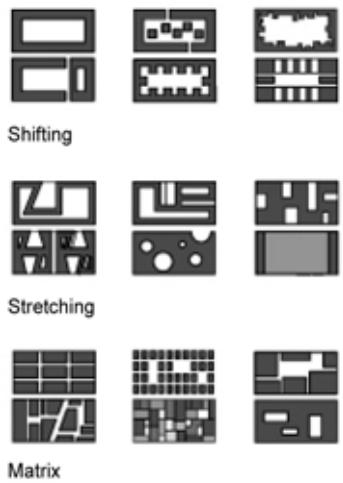


Figure 22
Block types in the IJburg plan, as defined in the architect's presentation schema.

21 This feature has already been implemented in Model B – see Chapter 7- CItyMaker

2 This step consists of defining a fixed number of block types. In principle the number of types is the same as the number of different labels given to the *islands* but the types already stored in a reusable library of block types can also be used. The block type definition uses a specific option for the UIP *DefineUUnit* called *BlockType* (see Table 6 and Appendix 2). This complex UIP requests the designer to specify the instructions that generate a particular block type and records the design sequence needed to generate it. The design sequence records the primitive block types and the operation used to produce the block type based on a predefined sequence of instructions: *Blt1*→*Blt2*→*Oper*, in which *Blt1* and *Blt2* are primitive block types, namely a closed block, linear block, punctual block (Pedro, 2001), matrix block (as found in IJburg plan – Figure 22) and an abstract block filling the entire *island*, and *Oper* is a binary operation in algebra U_{22} of two-dimensional shapes in a two-dimensional space following the definitions found in Stouffs (1994). The binary operations are sum (+), difference (–), product (·) and the symmetric difference ($\oplus$) – see Figure 24. However, symmetric difference is not used here because it needs more complex control mechanisms (Figure 27); the distance between buildings and building depth, for instance, becomes difficult to control. The same criterion is used in the application of more than one operation. This is actually the reason why symmetric difference is not being used – it corresponds to a combination of the operations sum and difference expressed by the equation $(A \oplus B) = (A - B) + (B - A)$. The formal exploratory potential increases but it becomes too difficult to control the generation in order to produce feasible results. Moreover, the operation product also presents particular difficulties involving similar problems. However, further work could be developed in this area. Some of the criteria established as generation goals can be determined in advance by the programme formulation as a set of requirements and constraints defined as description grammars and used to control the generation of designs. Additional heuristics may also help guide the generation towards suitable results. In any case, Figure 25 to Figure 28 show the potential of using multiple operations. Figure 25 also shows the potential of generating quadrants of a block separately, considering blocks subdivided into four equal quadrants defined by their symmetry axes. *BlockType* records all the requirements for each particular block type and the sequence of instructions needed to generate it. In fact, *BlockType* produces and records the specific discursive grammars needed to generate a specific block type. It may be said that this UIP generates customised UIPs which will later be incorporated into the generic code of the UIP *AddBlockType* (see UIP 029 in Appendix 2). So far, *BlockType* has proved capable of generating most of the types found in case studies 1, 2 and 3 (Figure 7)[22] . Skipping the details of the parameters involved in the operation, the spine-shaped block in Figure 20 (Rule 3)

22 This feature was implemented in Model A – see Chapter 7- CItyMaker – Figure 39 shows some results of the implementation of this pattern.

can be obtained from the sum of one linear block containing four columns of parallel buildings and a row consisting of one linear block containing one building. It should be noted that the density indicators can only be known in advance if a type is applied to blocks with the same size and proportion. If there are variations in size, the density indicators will also vary. This is easy to visualise in the case of the closed block. Managing morphological types in terms of density criteria using the density indicators involves strategic options and extensive calculations which require further research. However, it should be borne in mind that these parameters play an important role in the definition of heuristics used by *AddBlockType*, especially if density goals are set as design goals. Chapter 7 will discuss this subject further whilst addressing aspects of the implementations.

3 *AddBlockType* (which is an *AddUUnitbyLabel* option – see UIP 029 in Appendix 2) replaces a labelled island with a specific block type previously defined with *DefineUUnit*. The recorded block type is adapted to the size of the block which is being replaced. A set of constraints prevents the system from generating absurd results. To this end, designers must insert some control parameters into the system, recording for instance, the minimum and maximum building depth (set in terms of distinct orientations), minimum distance allowed between different buildings, minimum courtyard size and local access street width for the streets generated in the primitive matrix type. Different criteria can also be used for block type orientation when the *island* is replaced. The criterion used in implementation A (see Chapter 7) was to rotate the type towards the street with the highest hierarchy (see Figure 39). This feature is important when types are not symmetrical in relation to the *island* symmetry axes.

From this point on, we focus on the discursive structure of UIPs using the simple example of the UIPs that create a type coincident with one primitive block type, for instance, a closed block. This is a simplification of the pattern *AddBlockType* into a UIP containing a simple shape grammar consisting of rule schema (1) shown in Figure 21. This rule, which is a parametric rule, shows the available parameters working as variables that can be explored to produce design variations.

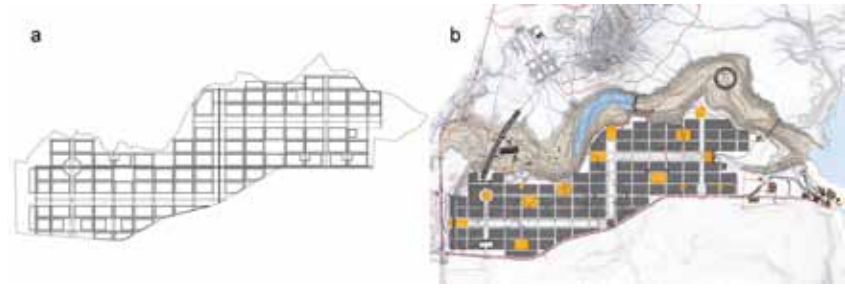


Figure 23
On the left (a): Design state after the generation of squares and before the generation of block typology. On the right (b): the same state in the architect's own drawing.

As previously stated, block types can have quite a high level of complexity and may involve several kinds of parameters and attributes. Parameters *h* and *w* are taken from the *island* being replaced whilst the other parameters are open variables, i.e. they are the open UIP options. The degree of freedom of the design is conditioned by the constraints developed upstream by the programme formulation, which may restrict the available range of values to a shorter interval or even further through direct designer input.

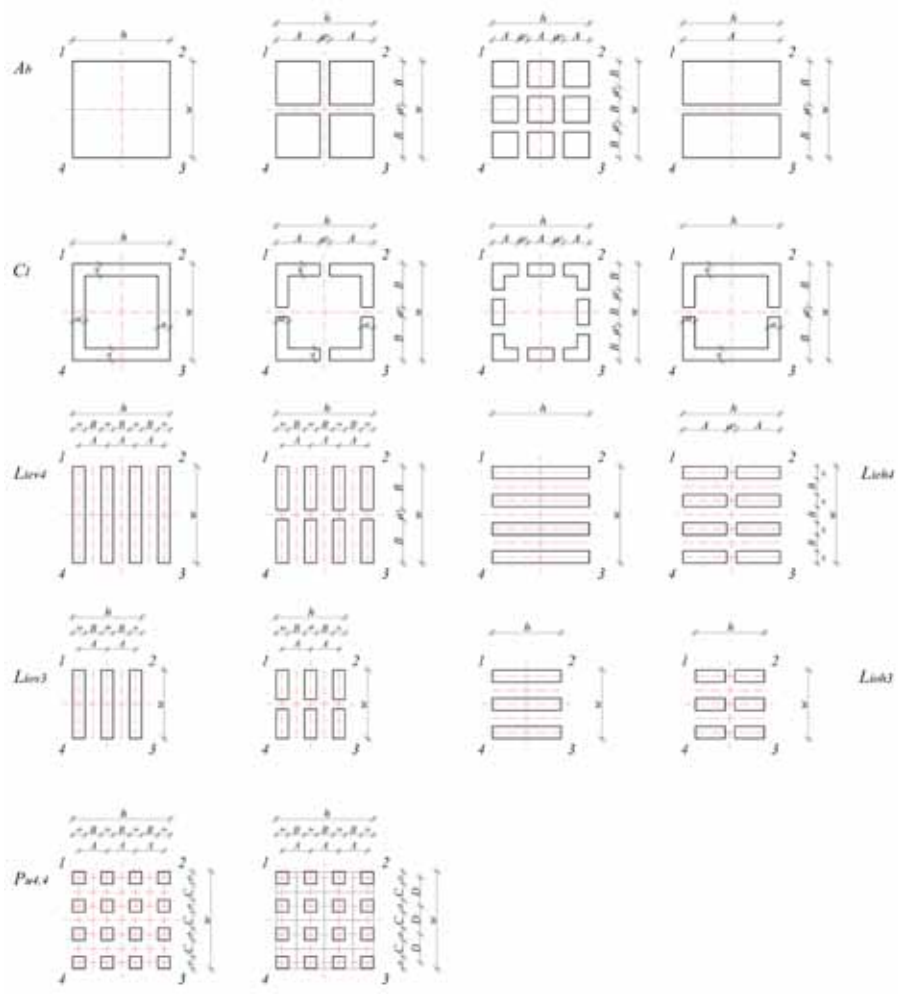


Figure 24
The 4 primitive block types: the island or abstract block (A_b), the closed block (C_l), the linear block (L_i) and the punctual block (P_u).

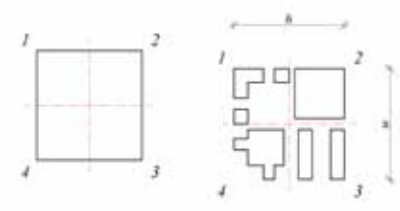


Figure 25
Block with 4 different quadrant configurations.

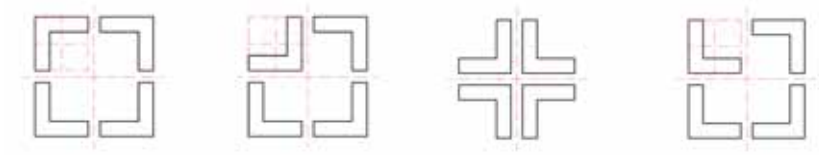


Figure 26
Transformation of a quadrant shape through a reflection of the shape using the 2 symmetry axes of the bounding box as the reflection axes. The second image shows a double reflection. The third shows the application of the same transformation to the 4 quadrants. The fourth shows the reflection of Q1 according to the horizontal symmetry axis.

Quantitative parameters can be framed within a range or specified by a function. Attributes are qualitative and change the meaning of features in the block type definition. The parameters can, for instance, be block width (*w*) and block length (*h*), building depth (*ta, tb, tc, td*), distance between internal facades (*ia, ib*), number of floors, etc. The variables in parentheses are the ones found in rule schema (1) in Figure 21. The attributes can be function, building type, or any kind of classifier relating a geometrical component to any non-geometrical feature. Other attributes concern block identification and placeholders for geo-referencing. Table 11 shows the parameters and attributes used in the block type in rule schema (1). Attributes that are not used in the example (e.g., function) were omitted in order to facilitate comprehension.

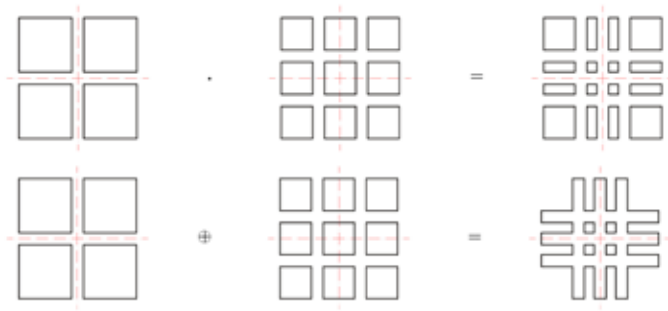


Figure 27
Illustration of product and symmetric difference operations. Although the operations show great composition potential they also show the difficulties inherent in controlling the meaning of the operations; for instance, controlling building depth or the distance between building façades.

At this point we turn to the discursive structure of UIPs. The generic structure of a UIP is defined as a discursive grammar of the form $\gamma_i = \{D, U, G, H, S_i, L_i, W, R, F, I_i\}$. In every discursive grammar, the set *G* contains the descriptions of the goals of the pattern defining its programming grammar and *D* contains the description rules for the generation defining the pattern's designing grammar. The description rules in *D* use the same features as the ones in *G* but are labelled with the letter *d* instead of *g*. In order to distinguish the description features of each UIP, the features in *ClassifyUUnitCells*, *ClosedBl* and *DefineUUnit* (*BlockType*) are labelled with the letters α , β and δ respectively.
In the BlockType application, the initial shape Ii is a labelled island in which the label corresponds to the variables identifier and classification in Table 11.

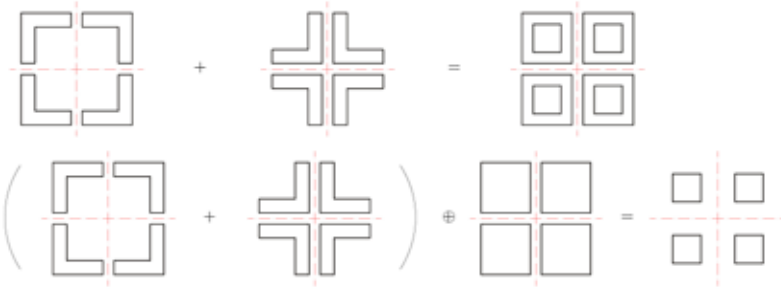


Figure 28
The operation sum and a compound operation involving the previous sum and a symmetric difference.

We now consider as the initial state of the design generation a design state starting with the classification of all *islands* after applying *ClassifyUUnitCells* – the initial state of the design is shown in Figure 29, page 168. The initial description *U* is defined through the descriptions α_{10} to α_{56} (see Table 9). *BlockType* can only be applied when these descriptions are found in the design. After the application of *ClassifyUUnitCells* all *islands* will have descriptions of the form:

$\alpha_{10} :< ObjectType, ObjectNumber, ClassName, Sublevels >$

Any particular island after the application of *ClassifyUUnitCells* will have the following format:

$\alpha_{10} :< island, \#, \mathbf{BL}, [UuBlt] >$

ObjectType is a particular instance of the class Block or **BL** (in this case an island), *ObjectNumber* is a counter (integer), *ClassName* is taken from the ontology (in this case **BL**), and *Sublevels* indicate the sublevel inheritance properties in the ontology (see Figure 9). *Sublevels* are future available parts of predicates. *Sublevels* may indicate an object class, meaning that all the objects in the class are available, or simply indicate specific object types within that class meaning that these types are the only ones responding to inheritance. For instance, if **BL** is indicated as a sublevel, any object in **BL** may be used by a rule. However, if *UuBlt* is indicated as a sublevel, only *UuBlt* object types can be used by future rules.

$\alpha_{20} :< h, w >$

$\alpha_{30} :< [classification, type] >$ where type corresponds to a *UuBlt* (a block type) and classification represents a label B_i where *i* marks a specific classification with a capital letter *A*, *B*, ... (it will be B_A in our example) identifying the classification previously set by the application of the pattern *ClassifyUUnitCells* (see figure in Table 9).The format will be:

$\alpha_{30} :< B_A, UuBlt >$

... placeholders which are geographic coordinates (α_{40} and α_{41}) and a record of the neighbouring street hierarchy (α_{42}). In addition, there are properties (α_{50} to α_{56}). α_{50} and α_{54} are fixed features of each *island*, whilst the others define the urban indicators to be achieved in the generation.

S_i is a set of parameterised blocks containing the 4 basic block types, the closed block, the linear block, the punctual block and the matrix block. This simplified version considers just the closed block - *ClosedBl*. *ClosedBl* is part of *AddBlockType*, as will become apparent later.

Set *R* contains the rules found in Figure 21 or, in this simplified version of the pattern, only Rule (1). It should be noted that the labels are omitted in this representation. However, the label B_i identifying the block classification determines one of the matching conditions for the application of Rule (1). In other words, if an island is marked by *ClassifyUUnitCells* as, for example, B_A, then only the block types defined by *BlockType* as type B_A can be applied to this island.

The *BlockType* pattern records the primitive blocks and operation used to generate a specific block type, as well as the parameters applied in each primitive block. *BlockType* involves only the features shown in Table 10. The behaviour of *BlockType* is synthesised

in the expression < *Blt1*, *Blt2*, *Oper* > where
$Blt1, Blt2 \in \{island, ClosedBl, LinearBl, PunctualBl, MatrixBl\}$ and $Oper \in \{+, -, \cdot, \oplus\}$. The generic descriptions of the primitive blocks are found in the ontology by requesting the pattern *AddBlockType*. Only *ClosedBl* is considered as an example here – see Table 11. *BlockType* records which primitive blocks are assigned to *Blt1* and *Blt2*, their respective parameters and which operation is assigned to *Oper*. There are only two possible design sequences, < *Blt1*, *Blt2*, *Oper* > or < *Blt1* > when *Blt2* and *Oper* are empty. The process is very interactive, as it basically requests input from the designer. *BlockType* involves only description rules defining design goals (from set *G*), therefore labelled with the letter g.
Rule g1 assigns a name to the block type, for example block type B_A, involving features δ_{10} and δ_{30}.

g1:

$\delta_{10} \leftarrow \delta_{10}$ $\quad - < island, \#, \mathbf{BL}, [?UuBlt] >$
$\quad + < island, \#, \mathbf{BL}, [B_A] >$

$\delta_{30} \leftarrow \delta_{30}$ $\quad - < ?classification, UuBlt >$
$\quad + < B_A, UuBlt >$, where *classification* is a user input.

Rule g2 inquires which primitive block should be assigned to *Blt1*.

g2:

$\delta_{60} \leftarrow \delta_{60}$ $\quad - < ?Blt1 >$
$\quad + < Blt1 >$,
$\quad ?Blt1, Blt1 \in \{island, ClosedBl, LinearBl, PunctualBl, MatrixBl\}$

In this example, the feature δ_{60} becomes $< ClosedBl >$. The next step requests parameters for the closed block (*ClosedBl*), which will be applied in the generation. It involves the parameters found in Table 11 which refer to the features of the closed block. Parameters h and w are not requested as they will be assigned later by the pattern *AddBlockType*.

g3:

$\beta_{21} \leftarrow \beta_{21}$ $\quad - < ?t_a, ?t_b, ?t_c, ?t_d >$
$\quad + < t_a, t_b, t_c, t_d >$, $t_a, t_b, t_c, t_d \in \circ \ \wedge \ \min_{bdepth} \le t_a, t_b, t_c, t_d \le \max_{bdepth}$
(defines building depths)

$\beta_{22} \leftarrow \beta_{22}$ $\quad - < ?i_a, ?i_b >$
$\quad + < i_a, i_b >$, $\quad i_a, i_b \in \circ \ \wedge i_a, i_b \ge \min_{inner}$
(defines minimum courtyard depths). By default $i_a, i_b = \min_{inner}$ defined in a table of standards.

Rule g4 requests which operation the user wants to use to generate block type B_A. In this example the answer would be none ($\varnothing$). However, the generic format of the rule is:

g4:

$\delta_{60} \leftarrow \delta_{60}$ $\quad - < ?Oper >$
$\quad + < Oper >$, $\quad Oper \in \{+, -, \cdot, \oplus, \varnothing\}$

Rules g5 and g6 are identical to Rules g2 and g3 but refer to *Blt2*. However, an initial conditional statement for the empty *Oper* jumps directly to Rule g7 which stores the recorded actions and parameters under the block classification B_A. This means that the block type B_A has the following description:

Variable description of the Urban Induction Pattern – *ClassifyUUnitCells*			
Features	**Sets of variables**	**Description notation α**	**Variables**
ID Object Identifier	Unique list per object	α_{10}	< *ObjectType*, *ObjectNumber*, *ClassName*, *Sublevels* >
Parameters	Block dimensions	α_{20}	< h , w >
Attributes	Typology: *Neighbourhood* *UuNei* *BlockType* *UuBlt* *Cluster* *UuClu*	α_{30}	< *classification*, *type* > where *classification* is marked with a label B_i and $i \in \{A,B,C,...\}$ *type* $\in \{UuNei, UuBlt, UuClu\}$
Geometry (coordinates)	Block corners Street nodes Neighbouring streets	α_{40} α_{41} α_{42}	< Bl_1 , Bl_2, Bl_3, Bl_4 > < 1 , 2 , 3 , 4 > The elements in the two sets,{ Bl_1 , Bl_2, Bl_3, Bl_4 } and { 1 , 2 , 3 , 4 }, represent geographic coordinates < s_a , s_b , s_c , s_d > , records the street hierarchy of neighbouring streets
Properties	Area-island (A_i) Building intensity (*FSI*)* Coverage (*GSI*)* Spaciousness (*OSR*)* Area-cell (A_c) Building height (*L*)* Maximum Nr of Floors (*NF*)	α_{50} α_{51} α_{52} α_{53} α_{54} α_{55} α_{56}	< A_i > a real number expressed in unit m^2 < *FSI* > a real number expressed in unit m^2/m^2 < *GSI* > a real number expressed in unit m^2/m^2 < *OSR* > a real number expressed in unit m^2/m^2 < A_c > a real number expressed in unit m^2 < *L* > where *L* is a real number indicating the average nr of floors < *NF* > where *NF* is an integer indicating the number of floors

* Properties follow the notations and definitions proposed by Berghauser-Pont and Haupt (2010). The block properties are calculated at block or *island* level. In order to avoid unit conversion the areas are expressed in m^2 instead of hectares.

Table 9
ClassifyUUnitCells – Urban Unit variables

δ_{10} : < *island*, #, B_L, [B_A] >
δ_{30} : < *BA*, *UuBlt* >
δ_{60} :< *Blt1*,∅,∅ >
where *Blt1=ClosedBl* and has the features $\theta_{21} < t_a, t_b, t_c, t_d >$ and $\theta_{22} < i_a, i_b >$, all variables being defined by user input.

Blt1 contains the descriptions that will be assigned to *AddBlockType* for the generation of the final block type. In detail, considering this simplified example, $Blt1 = [ClosedBl, (t_a, t_b, t_c, t_d), (i_a, i_b)]$. The generic format describing *Blt1* and *Blt2* is $[blocktype, \theta_{21}, \theta_{22}]$ where θ_{21} and θ_{22} represent the descriptions to assign to features θ_{21} and θ_{22} of *AddBlockType* when generating *Blt1*. The full description of the design sequence in the example of Rule (1) (Figure 21) is
$\delta_{60} =< [ClosedBl, (t_a, t_b, t_c, t_d), (i_a, i_b)], \emptyset, \emptyset >$.
AddBlockType automates the generation of block types. Its algorithm identifies *islands* and the classification B_A attributed to the *islands* which are the matching conditions (g1), extracts the required data from the existing *island* (g2) according to the classification it identifies: the block type to apply and its respective design sequence (g3), the generic parameters to be applied to the first block type in the design sequence – *Blt1* – (g4), the generic parameters to be applied to *Blt2* – (g5), and the operation, *Oper*, to be applied to *Blt1* and *Blt2* (g6). The last rule erases the block type label B_A (g7).
In *AddBlockType* set *G* contains the following description rules defining the goals to be achieved in the generation of this pattern:

Rule g1 – a matching rule: checks conditions in the existing *island*.

α_{10} :< *ObjectType, ObjectNumber, ClassName, Sublevels* >
$ObjectType = island \wedge ClassName = \mathbf{BL} \wedge Sublevels \subset B_A$
α_{30} :< *classification, type* >
$classification = B_A \wedge type = UuBlt$

Rule g2 – the initial state: extracts data from the existing island and adds the applicable parameters to the descriptions. The data is extracted from the geometry, from the tables of requirements (Table 7 and Table 8), or from previously driven data recorded in the generation model's database.

$\theta_{10} \leftarrow \alpha_{10}$	– < *island*, #, B_L, [B_A] > + < B_A, #, B_L, [P_L, B_D, L_U] >
$\theta_{20} \leftarrow \alpha_{20}$	i.e., < *h*, *w* >
$\theta_{21} \leftarrow \emptyset$	
$\theta_{22} \leftarrow \emptyset$	
$\theta_{30} \leftarrow \alpha_{30}$	i.e., < *classification*, *type* > and in this case < B_A, *UuBlt* >

$\theta_{40} \leftarrow \alpha_{40}$ geographic coordinates for points <Bl_1, Bl_2, Bl_3, Bl_4>

$\theta_{41} \leftarrow \alpha_{41}$ geographic coordinates for points <*1*, *2*, *3*, *4*>

$\theta_{42} \leftarrow \alpha_{42}$ indicates street hierarchy of neighbouring streets $<s_a, s_b, s_c, s_d>$

$\theta_{50} \leftarrow \alpha_{50}$ area extracted from geometry <A_i>

$\theta_{51} \leftarrow \alpha_{51}$ indicator derived from table of requirements <*FSI*>

$\theta_{52} \leftarrow \alpha_{52}$ indicator derived from table of requirements <*GSI*>

$\theta_{53} \leftarrow \alpha_{53}$ indicator derived from table of requirements <*OSR*>

$\theta_{54} \leftarrow \alpha_{54}$ area extracted from geometry < A_c >

$\theta_{55} \leftarrow \alpha_{55}$ indicator derived from table of requirements <*L*>

$\theta_{56} \leftarrow \alpha_{56}$ – < ?*NF* >
+ <*NF*>, an integer from user input or imported from the ontology based on programmatic definitions or available upstream regulations

$\theta_{57} \leftarrow \varnothing$

$\theta_{60} \leftarrow <>$

Rule g3 – imports the design sequence defined from user input using *BlockType* (feature δ_{60} and assigns it to θ_{60}).

$\theta_{60} \leftarrow \delta_{60}$ $- <?Blt1, ?Blt2, ?Oper>$
$+ <[ClosedBl, (t_a, t_b, t_c, t_d), (i_a, i_b)], \varnothing, \varnothing>$

This information allows rule g4 to assign the parameters recorded in the block type definition to the features θ_{21} and θ_{22}, thereby defining the design goals for building depth and courtyard measures.

Rule g4 – assigns parameters to a closed block:

$\theta_{21} \leftarrow \theta_{21}$ $- <?t_a, ?t_b, ?t_c, ?t_d>$
$+ <t_a, t_b, t_c, t_d>$ where $t_a, t_b, t_c, t_d \in Blt1$ extracted from *Blt1* position in θ_{60}.

$\theta_{22} \leftarrow \theta_{22}$ $- <?i_a, ?i_b>$
$+ <i_a, i_b>$ where $i_a, i_b \in Blt1$ extracted from *Blt1* position in θ_{60}.

... and defines a temporary label for *Blt1*:

$\theta_{30} \leftarrow \theta_{30}$ $- <B_A, ?type>$
$+ <B_A, Blt1>$

The temporary label allows the algorithm to distinguish between the values assigned to parameters in *Blt1* types and values assigned to parameters in *Blt2* types. The complete set of design instructions considers the generation of a block type involving two primitive block types. Rule g5 executes the same procedures as Rule g4 for extracting parameter goals for *Blt2*. A single rule, g6, sets as a design goal the

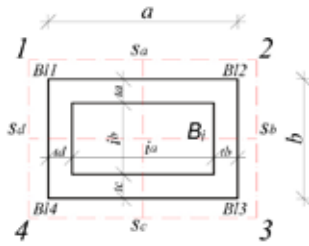
application of the operation defined as *Oper* in θ_{60}. The last Rule, g7, erases the temporary labels *Blt1* and *Blt2* and the block type label B_A.
The designing grammar in *AddBlockType* includes Rule (1) from Figure 21 and the description rules (d) from set *D* which essentially assign the parameter values and instructions captured by the rules (g) of the programming grammar to apply in the application of the shape rule (1). In other words, for each rule g1 to g7 in set *G* there is a rule d1 to d7 in set *D* which assigns the parameter values for the instantiation of shape rule (1). Rules d1 to d5 generate first *Blt1* and then *Blt2*. Rule d6 applies the defined operation (*Oper*) between *Blt1* and *Blt2*, and Rule d7 erases all labels assigned to the block. In the case of $Oper = \varnothing$ the label *Blt1* is erased, thereby ending that particular block design and proceeding to Rule d7 by erasing the label B_A.
The algorithm is recursive. The generation of block types starts with Rule g1 by checking the existence of label B_i classifying *islands* and then proceeds to Rule g7. It then generates the corresponding block type using the local *h* and *w* parameters (block size) applying all the design rules d1 to d7. The first rule g1 searches again for the matching conditions in the design. The generation stops when all B_i labels are erased.

Variable description of the Urban Induction Pattern – *DefineUUnit*		
Features	**Description notation δ**	**Variables**
ID Object Identifier	δ_{10}	< *ObjectType*, *ObjectNumber*, *ClassName*, *Sublevels* >
Attributes	δ_{30}	< *classification*, *type* > where *classification* is marked with a label B_i and $i \in \{A,B,C,...\}$ *type* $\in \{UuNei, UuBlt, UuClu\}$
Design sequence	δ_{60}	< *Blt1*, *Oper*, *Blt2* > where $Blt1, Blt2 \in \{island, ClosedBl, LinearBl, PunctualBl, MatrixBl\}$ and $Oper \in \{+,-,\cdot,\oplus\}$. *Oper* and *Blt2* can also be $\varnothing$.

Table 10
DefineUUnit – Urban Unit variables

The shape set S_i in γ_i contains the primitive block shapes and the rule set *R* contains the shape rules shown in Figure 21. The reader is invited to infer the description rules for any application of the UIP *AddBlockType* including shape rules (2a), (2b) and (3) in Figure 21or any other block type resulting from a design sequence of $<Blt1, Blt2, Oper>$.
Set *F* contains the functions that establish relationships between some of the features describing the urban blocks; for instance, the functions that allow urban indicators to be calculated are defined in this set. More precisely, such functions update properties in the design.

Further details regarding the design workflow and the standard application of *AddBlockType*, rather than just one of its rules are needed to explain the use of *H* (heuristics) and *W* (weights) sets. So far, a design workflow has been presented that is based mainly on the designer's interaction with the design tool, noting sometimes that some of the input may come from the programme formulation. Regardless of how the table of requirements (Table 7) is completed, there are considerable differences in the character of the urban environment defined by applying different block types. In this regard, the exploration and distribution of typology could follow specific heuristics or rules of thumb to achieve specific types.

BL – Blocks	Sets of variables	Description notation β	Variables
A1 – Closed block	**ID**		
 The label B_i represents the variable *classification*	Object Identifier	β_{10}	< *ObjectType*, *ObjectNumber*, *ClassName*, *Sublevels* >
	Parameters		Parameters*
	Block dimensions	β_{20}	< h , w >
	Building depth	β_{21}	< t_a , t_b , t_c , t_d >
	Inner courtyard dimensions	β_{22}	< i_a , i_b >
	Attributes		**Attributes (labels)****
	Block type and Block classification	β_{30}	< *classification*, *type* > where *classification* is marked with a label B_i and $i \in \{A,B,C,...\}$ and *type* $\in \{UuBlt\}$
	Geometry		**Geometry (coordinate placeholders) (labels)****
	Block corners	β_{40}	< Bl_1 , Bl_2 , Bl_3 , Bl_4 >
	Street nodes	β_{41}	< 1 , 2 , 3 , 4 > The elements in the two sets, { Bl_1 , Bl_2 , Bl_3 , Bl_4 } and { 1 , 2 , 3 , 4 }, represent geographic coordinates < s_a , s_b , s_c , s_d > , records the street hierarchy of neighbouring streets
	Vicinity - streets	β_{42}	
	Properties		**Properties**
	Area-island (A_i)	β_{50}	< A_i > a real number expressed in unit m^2
	Building intensity (*FSI*)***	β_{51}	< *FSI* > a real number expressed in unit m^2/m^2
	Coverage (*GSI*)***	β_{52}	< *GSI* > a real number expressed in unit m^2/m^2
	Spaciousness (*OSR*)***	β_{53}	< *OSR* > a real number expressed in unit m^2/m^2
	Area-cell (A_c)	β_{54}	< A_c > a real number expressed in unit m^2
	Building height (*L*)***	β_{55}	< *L* > where *L* is a real number indicating the average nr of floors
	Maximum Nr of Floors (*NF*)	β_{56}	< *NF* > where *NF* is an integer indicating the number of floors
	Footprint (B_d)	β_{57}	< B_d > a real number expressed in unit m^2
	Design sequence		**Design sequence**
		β_{60}	< *Blt1* > where *Blt1* is a *ClosedBl*

* It should be noted that a few more description notations should be added in the case of the linear block, the punctual block and the matrix block.

** Attributes and geometry placeholders are the only variables represented with labels in the grammars. Eventually all variables can be labelled if any variable is needed for the definition of particular rules in the grammar.

*** See note in Table 9

Table 11

AddBlockType – Closed block features and variables. Similar tables of features apply to the other primitive types.

A similar comment can be made regarding the issue of density in relation to urban morphology. The two subjects are actually related, as can be seen in Spacematrix (Berghauser-Pont and Haupt, 2010). However, this theme is problematic because it involves subjectivity, which should be addressed by the designer. For instance, it can be easily understood from the model picture (Figure 30) that in Plan 1 the designer is trying to place specific block types next to the main streets. More precisely, closed blocks are placed along the central promenade and spine-shaped blocks are placed along the two parallel promenades. Instead of the designer intervening directly, this behaviour can be induced directly into the UIP by defining the heuristics that establish the conditions that distinguish which type to apply, depending on the position of the block type in the design. Weights could be provided to relate type to position or density distribution in order to manage building height. Although this kind of decision is seen clearly in the model – the buildings facing the promenades have two extra floors – the generation of these details involves additional rules which are set as options in the UIP *ManageBuildingHeight*. However, the most important aspect is to allow for the system to provide as much user interactivity as possible in order to offer some customisable features. The important aspect to stress here is that this mechanism allows a set of optional behaviours to be built into this UIP which can be based on calculations derived from the density indicators, thus introducing design decisions based on a 'spacematrix' interpretation of available or user defined patterns. We will return to this subject in the discussion section.

Figure 29
Plan layout including the labelled islands after classification with ClassifyUUnitCells.

§ 6.4 Storing and using design data

Two main difficulties had to be overcome in developing CItyMaker. The first was to define a system which would allow for creative design synthesis whilst maintaining standard reflective design practice, and the second involved data manipulation and data-based decision-making in the design process, as well as assigning generated data to existing or generated representations. In addition, all generated representations had to be generated in GIS-compatible formats in an appropriate layered structure.

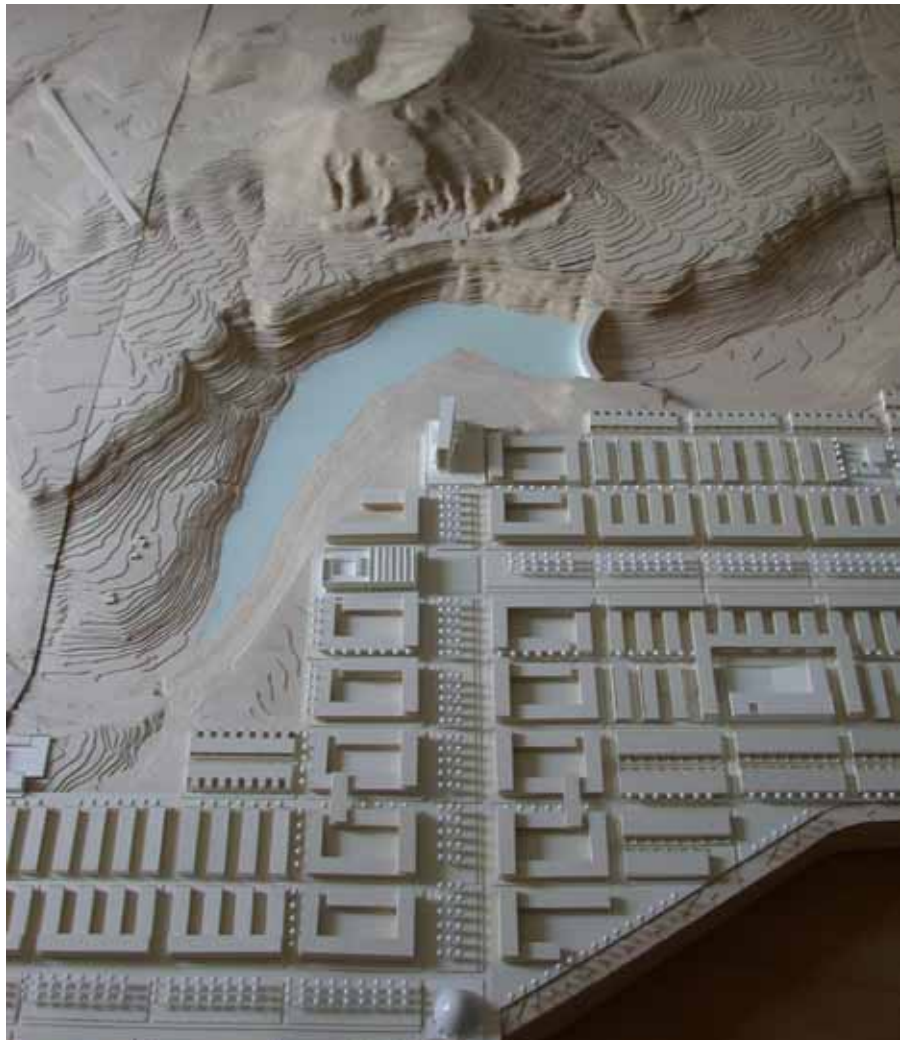


Figure 30
The Praia plan model showing the closed blocks along the main promenade.

The ultimate goal of CItyMaker is to generate meaningful urban designs. However, computing meaningful designs involves conceptual difficulties of a semantic nature. Shape grammars, even when fully detailed, are not able to generate meaningful designs or, to put it more accurately, find the appropriate designs in the language for a particular context. The Prairie House grammar (Koning and Eizenberg, 1981), which is a very accurate grammar in terms of definition of language, does not contain a means for deciding which rule to apply in a particular context or the appropriate parameters for the design context. Although the designs in the solution space do fit the language of Frank Lloyd Wright there is no way of choosing which rules to apply at each step of the derivation process except through designer input. In an automated generative process an additional formalism is needed for this purpose.

Stiny (1981) introduced description grammars to solve this problem. Description grammars provide rules relating other design features to the shapes in the design. With this formalism uses can, for instance, be assigned to rectangles representing buildings and consequently particular characteristics can be assigned to buildings according to their function and shapes can be transformed accordingly.

Duarte (2001) improved this formalism by introducing heuristics to guide the generation towards given goals. The heuristics introduce some 'rules of thumb' relating formal options, for instance, to priorities in design requirements. In the case of Duarte's Malagueira grammar, there is a specification that gives priority to certain proportions in the rectangle subdivision rules. However, in the previous (Prairie House and Malagueira House) grammars there is no real creative activity in the process and it can scarcely be called design. The design systems are in fact search systems which try to find adequate complex architectural solutions in a vast design space bounded by a predefined (or pre-designed) design language. The language is given; it is not the product of a process of synthesis.

Synthesising a design language is a much more complex problem because it involves no knowledge of the design rules in the initial states of the design process. The traditional method of designing evolves through progressive steps (moves) of trial and partial solutions which are critically assessed before advancing to another move. Decisions, as well as decision-making criteria, are applied locally at each step in the design process and knowledge of the design problem evolves with the design. The main idea developed for CItyMaker applies the discursive grammar formalism to these local levels, i.e. to generic design moves. Creative design is therefore still possible following a standard reflective procedure, but the search is enhanced with the generative properties of design moves. This concept solves the problem of making creative design synthesis possible in CItyMaker. It also partially solves the generation of GIS compatible formats and data manipulation. However, in order to fully solve the two main problems as well as the layered representations, the ontology plays an essential role.

The ontology organises all the concepts involved in the description of the urban environment into a relational / hierarchical structure – see Figure 9. The ontology presents a structure of related concepts forming thematic systems to describe particular aspects of the urban environment:

1 The city as a street system;
2 The city as a built system
3 The city as a property system
4 The city as a natural system
5 The city as a reference system
6 The city as a public space system
7 The city as an activity system
8 The detail system (defining the city's character)

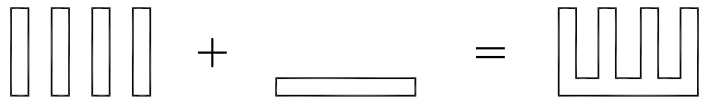
Each system (theme) is composed of several interrelated object classes in similar ways to the one showed in Subsection § 5.4 , page 92, where the 'city as a street system' is explained. Explicit relationships exist between the concepts of one system and the concepts of another, and there are primary and secondary relationships between classes, thereby introducing two levels of priority in inheritance relations. What the relationships predefine is part of the predicate conditions of patterns and some of their consequents. In other words, a pattern (UIP) can only be applied if the objects in the predicate belong to the object class that it is supposed to generate, or belong to their immediate upstream related classes. Similarly, UIPs generate designs only from objects belonging to the initial shape structures and/or from immediate downstream related classes. In both cases, primary and secondary relationships are considered and priorities established based on this criterion.
As an example of the use of primary and secondary relationships, patterns generating grids also generate a structure of blocks defined in the **BL** class. Blocks (**BL**) are primarily developed into buildings (**BD**) and block parcels (**BLP**) into building plots (**BP**). However, depending on other features of the context, blocks can be transformed into squares (**SQ**). In such cases, the patterns transforming blocks into squares use the secondary relationships.
The relational structure of concepts (classes) provides two important features of the system:

1 The inheritance structure in the design system specifies the design moves allowed in each step of the design generation process,
2 The classes provide the layered structure. Each class generates objects belonging to specific layers according to the object types they generate and the particular type of geometry that makes up their representations (points, lines or polygons).

Consider the pattern *AddBlockType*. In the initial shape set (*Ii*) the UIP contains the primitive shapes needed to generate block types and in the rule set (*R*) the

transformations needed to compose two primitive shapes into a specific block type. If a type is generated by adding a set of linear parallel buildings to another to form a spine-shaped block as shown in Figure 21, Rule 3, the algorithm uses a primitive linear block with 4 parallel buildings and adds a perpendicular linear block with only one building using the operation addition[23]:

Liev4 + Lieh1 (see Figure 24)



The ontology contains the definitions of the primitive blocks in the Blocks class **BL**, in this case the linear block, involving the variables and parameters shown in Figure 24. Some parameters may already be provided by the system – for instance, w and h are previously extracted from the island (rule g2: $\beta_{20} \leftarrow \alpha_{20}$) – and others may be provided by user input or from existing functions or heuristics. As an example, if the linear block is chosen, a function might define the number of parallel buildings so that they always have a space between them that is larger than the user input (or regulation) defining the minimum distance allowed between two buildings. A heuristic might state that if the generation is conditioned by proximity to a main street (defined by a set of weights), (1) the block is rotated so that the continuous side of the spine faces the main street – see implementation A, Figure 39 – and (2) the building height is raised by one floor in the primitive type *Lieh1*, i.e. if *Liev4* has, for example, 2 floors, *Lieh1* is raised to 3 floors. This heuristic generates the spine blocks as defined in Plan 1 – see the model photo in Figure 30. In practical terms, the definitions of primitive blocks can be stored in the ontology as code including definitions of the primitive shape – geometry – and descriptions which include requests for the variables and parameters needed to instantiate a particular block. Therefore, the main code in CItyMaker searches the ontology each time a primitive type is used. Conversely, similar definitions (primitive object types) may be found in other classes in the ontology[24].
Height management in the system uses the UIP *ManageBuildingHeight*. *ManageBuildingHeight* is applied to the stored block types, applying a user defined number of floors to the primitive blocks. The first primitive block uses an input integer and the second is defined through a maximum difference between *Blt1* and *Blt2*, also in terms of the number of floors. The generation of block types also requires some complex heuristics to manage the height differences between the two primitive blocks

23 In implementation A the designer has the option of choosing the position of the building in the case of a linear block containing only one building – the designer chooses either one of the sides or a centred position.

24 Only partially implemented. These features need to be implemented in collaboration with the formulation model.

in order to guarantee the generation of meaningful blocks. The height information is stored with the block type and applied in the generation of block types with *AddBlockType*. If the stored block type does not contain this information the pattern *ManageBuildingHeight* is activated to gather the missing information.

The management of building height is a subject directly related to the management of plan density. It may be said that it defines a strategy linking construction footprint and building height. Due to the development of Model B (see the next chapter) in which density management is accomplished at district level leaving blocks with generic regulations for future design, research into the management of building height was left incomplete[25]. However, the main idea was to control the distribution of density using definitions of building types and attraction functions similar to those in Model B, which define a relationship between distance to the main urban features (main streets, main squares, centre) and block type. Two types of functions should be used to control density distribution behaviour in the design system. First, this involves the relationship between types and density, for which spacematrix research (Berghauser-Pont and Haupt, 2010) would provide some insights. Basically, certain heuristics would control which types are suitable for specific density measures. Secondly, the height of block types would also be affected by density goals. In this case, depending on the density goals, a function would change the block type definitions to fit the required density, which would always be constrained within meaningful boundaries. These concepts should be improved in future work.

The parallel grammar structure of the system separating grammar by class and geometry type (points, lines or polygons) guarantees the generation of structured representations that are appropriate for geographical information systems. The ontology, whilst providing the inheritance structure, also guarantees a correct procedural structure for the system, i.e. it only allows UIPs to be displayed in a semantically correct order. Furthermore, the ontology plays an essential role in data-based decisions and data generation. In fact, all the concepts in the ontology contain formal descriptions of the objects in each class. Properties are also included in these descriptions (see Table 9 and Table 11) containing measures and/or density goals. Table 9 shows the properties involved in the pattern *ClassifyUUnitCells*, and Table 11 shows the properties involved in the pattern *AddBlockType*. As such, the former refer to density goals and the latter to density measures.

To explain further, *ClassifyUUnitCells* is a pattern which allows for the semantic classification of urban units (in this case the units are blocks). The classification uses semantic information stored in the ontology, such as the street hierarchy, existing focal points and weights attributed to these elements. The system maps the weighted

25 In Model B we agreed that planning on a district scale would end in a design layout containing regulations for block design rather than specific morphological types. This would leave scope for block design constrained only by generic regulations essentially based on density indicators. This is also consistent with common practice in terms of phase subdivision for large scale plans. This practice is clear in case studies 3 and 4 – IJburg and Ypenburg – see Figure 7.

elements defining an attraction 'force field'. By default ClassifyUUnitCells distributes a user-defined number of block types, for example types {*A, B, C, D*}, assigning the denser ones to the areas with higher attraction. In this process, the density goals of the plan, previously defined in the tables of requirements (see Table 7 and Table 8) or by the programme formulation, are distributed to the islands in the grid using bounded distribution methods. Consequently, this distribution defines an uneven distribution of density per island which can be understood as occupation regulations for that island, expressed in terms of density measures defined at island level. The density properties in *ClassifyUUnitCells* correspond to density goals for block generation. *AddBlockType* uses these density goals, applying them to the assigned block type and generates the type according to the goals. The block type generated also has associated properties, as shown in Table 11 but this time they are facts, i.e. they are the density measures of blocks.

In addition to the hierarchical structure defined by the explicit relationships between classes, the ontology also allows the rules defining conditional relations between the involved parameters to be edited. The ontology rules contain conditional statements. The semantic control of the design system is, therefore, divided by two controlling devices: the ontology and the UIP discursive formalism, which have different roles. The ontology controls generic and abstract relationships between concepts and representations describing the urban environment which are independent of the creative design process – e.g. regulations and design standards. The UIPs control meaningful relations which are intrinsic to the design process, such as the synthesis of a design language and achieving design goals. The ontology and programme formulation is implemented by Montenegro et al (2011) incorporating these concepts and others exclusively related to programme formulation. The outputs of the programme formulation correspond to most of the user inputs needed at the start of the generation process, for instance, data requested by the tables of requirements (Table 7 and Table 8) and a few geometrical inputs. Even references (R_{ef}s) and respective weights can be suggested by the programme formulation. Nevertheless, it was always the policy of both researchers to allow the designer the possibility of changing any of these definitions. This detail makes the tools extremely powerful, both individually and collectively. The main reason for including this editing feature is not simply to offer design freedom at whim, but is also a necessity if the design tool is aiming for universal usability. The impossibility of foretelling all possible design situations from all possible contexts creates the need for a user edit feature and customisation of the design system. CItyMaker is therefore an autonomous design tool for designing urban plans that includes an already vast set of assessment interfaces based on density indicators which help users determine whether their decisions are reaching the goals.

§ 6.5 Conclusion

CityMaker is an urban design generation tool which can be used autonomously as a design tool or as part of the City Induction design system in cyclical workflows of analysis – formulation – synthesis – evaluation. This workflow can be summarised in the following diagram:

The main knowledge contributions found in this chapter are:

- a design method;
- a semantically controlled generative design formalism;
- an ontology of the urban design process;
- a system generating designs in GIS compatible formats;
- a grammar-based design system providing a means for developing customised design languages.

This chapter starts with a description of a design workflow or method to be applied in the urban design process using the generative tool described in this Chapter. The workflow is suitable for a complete urban design process, starting with participatory decisions developed in the pre-design phase and evolving interactively until the goal-based design solutions are produced. Given that practice tends to separate the vision development, programme formulation, design synthesis and evaluation processes, CityMaker was developed to work as far as possible without the need for any of the specific City Induction tools used upstream or downstream in relation to the workflow showed above. In order to do so, the tool provides an interface for typical input data encompassing geometrical and geo-referenced information, generic goal inputs based on density indicators, and an interface for urban plan outputs defined in terms of a morphological output (a layout) and the corresponding density-based indicators. It may therefore be said that CityMaker already displays the information needed for a visual but integrated evaluation of designs, including some performance indicators. The main contribution in terms of design methods is the possibility of assessing layouts and corresponding density indicators move by move, enhancing the designer's awareness of the consequences of his/her design moves.
Two devices – the ontology and the discursive UIP formalism – are proposed to control design semantics. Description grammars guide the design rules towards meaningful

design moves. Similarly, the information generated in a design move provides guidance for the application of the next design moves. The matching conditions of a UIP (the predicate) are defined through a set of descriptions and, as part of the consequent, the UIP also produces descriptions of the minimum set of design features that need to be recognised for the application of other UIPs. The patterns that match such conditions are the patterns available in the next step of the generation. The ontology provides most of this information and also the heuristics guiding the rules. The design system provides customisable features, the most important of which is the possibility of customising specific UIPs from a generic UIP structure. The example provided shows that the system can record a set of custom operations which, with the addition of the common pattern structure, can create a specific customised UIP to be stored in a personal library of design patterns for reuse in any generation context.

7 CityMaker – Implementing Urban Grammars – two prototypes

The previous chapters described a theoretical model for developing a generative urban design system based on the generative properties of shape and description grammars. It aims to produce flexible urban systems in response to the need to improve flexibility in city planning. At the end of the design process the designer obtains a design language in the form of an urban grammar defining the flexibility space of the design and an illustrative layout. The generative urban design system allows for dynamic interaction between design exploration and data flow on density-based indicators in order to support design decisions and improve design quality. The system also aims to respond to input needs and provide dynamic information on the measurements and properties of the solutions being explored. The idea is that the quality of design decisions improves with the quality of the information flow. Methodologically speaking, integration with GIS environments is considered essential in order to accomplish this, as most procedures associated with pre-design analysis and design evaluation are performed in GIS platforms. In order to achieve the desired interactivity the generation system needs to:

- easily input programmatic data resulting from analytical procedures performed in a design context;
- dynamically provide data output for measurements and properties of plans, including density measures during the design exploration process, and dynamic comparison with the programme design goals;
- allow the design goals to evolve dynamically in response to design exploration and the respective data feedback;

Technically this means that the system needs to:

- easily communicate and retrieve data from analytical and programming tools either by designer input or, preferably, by direct input from such tools, depending on the analysis;
- easily communicate and dynamically feed data into analysis and evaluation tools.

In the context of the City Induction research project, dynamically integrating these behaviours into formulation, generation, and evaluation is a main goal. By analogy with the concept of building information modelling (BIM), the project aims to develop the foundations of CIM, i.e. city information modelling, a tool for designing and supporting urban design decisions. Such a tool is also believed to provide a good platform for monitoring design implementation due to its integrated formulation, generation and

evaluation features as well as its capacity to update designs and information, but this, of course, needs specific applied research. However, from the point of view of an isolated generation and design exploration tool – CItyMaker – the most important aspect is that the system dynamically provides as much information on evolving design as possible, together with efficient visualisation of the evolving designs. Urban designers are usually quite proficient in interpreting data but due to human limitations in dynamically processing large amounts of data (Simon, 1981) it is not an easy task to process complex calculations whilst designing. The main idea is to be able to provide calculations move after move whilst designing, thus consolidating ideas in an iterative process (Moughtin, 2000).
In this chapter, two different prototype implementations are presented and compared. The first implementation is more concerned with the correct generation of representations and data in GIS compatible formats as a means of promoting an integrated design workflow. The second implementation focuses on designer interaction, usability and real time dynamic data access. The first implementation will be called Model A and the second Model B. Generically speaking, the implementation of the City Induction generation model is called CItyMaker.
This chapter is divided into six sections. Section a reports on the survey of software platforms carried out prior to the start of the first implementation (Model A). Section b presents Model A. Section c starts by explaining how a shape grammar model is converted into a parametric design model and then presents Model B. Section d compares the two models, highlighting the qualities and shortcomings of each model. Section e develops the discussion, providing some insight into the validity of the approach focusing essentially on the validation of the theoretical model. The concluding remarks in Section f offer some recommendations for the development of CIM software.

§ 7.1 Existing tools and their limitations: CAD-GIS platforms and Shape Grammar interpreters

Before starting the description of the implementations, this subsection surveys the existing tools that could be used to implement a generative urban design system. It does so from two different perspectives: the perspective of the City Induction research project in which integration of CAD and GIS interoperability is the main criterion, and the isolated perspective of the generation module which is envisaged as an autonomous urban design tool and needs to integrate algorithmic features, in particular a shape grammar interpretation environment. As such, this subsection is divided into two parts: the first considers the survey of a common (City Induction) design platform, and the second the survey of shape grammar interpreters and their connection with the project's common design platform.

§ 7.1.1 Survey of a common City Induction design platform

It was implicit in the City Induction *project* (see the end of Section § 1.4 - page 29) that to achieve the project's main goals we would need to combine (at least at generation level), Computer Aided Design (CAD), Geographical Information Systems (GIS) and Shape Grammar software. In order to make a decision on this matter, the research team carried out a survey of the existing tools.
Computer Aided Architectural Design (CAAD) software is well established within architectural practice and teaching, offering a wide range of tools for designing, drafting and modelling buildings. However, when it comes to dealing with urban design these tools do not seem to address most of the information regarding the urban environment adequately, as they lack the ability to manage the contextual and site information needed to support and describe an urban plan. Common CAD tools include software such as AutoCAD, Microstation, ArchiCad, Rhinoceros and Vectorworks.
On the other hand, Geographic Information Systems (GIS) are able to manage huge amounts of geographical information in different formats – representational, either raster or vector, and linked data stored in several databases - linked by a common geographical position. However, they lack the design features and tools necessary for a creative design process. ArcGIS, MapInfo, Manifold, BentleyMaps and AutoCADMap are some of the most common GIS software packages.
In order to integrate programme formulation, design generation and urban evaluation in a single design tool a shared analysis and design platform which can perform these three activities with maximum possible interoperability is needed. The City Induction research team started with the main consensus that the shared platform should integrate CAD and GIS capabilities in the best possible way. Although, there was no completely integrated platform offering the necessary interoperability between CAD and GIS capabilities, the capacity to offer features from both types of software was the basis for defining a shortlist of platforms to select as candidates for supporting the City Induction implementation/s. The shortlist included the following software:

1. CityCAD version 1, by Holistic City [WS5]
2. CityEngine 2009, by Procedural [WS6]
3. CityZoom, by SIMMLAB [WS7]
4. AutoCAD Civil 3D 2009, by Autodesk [WS8]

CityCAD is a Microsoft Windows tool launched in June 2008, developed by the company Holistic City. Departing from a design oriented CAD paradigm, it focuses on the development of an interactive interface for urban design, incorporates an urban design ontology with attributes assigned to the design entities, and reports on a variety of design data analysis (Holistic City Software 2010; de Boer, 2009). Version 2.0 was

launched with new features soon after this survey was completed. The tool does not provide any programming interface for developing application extensions.
CityEngine is a Microsoft Windows, Mac OS X and Linux tool stemming from research originally conducted at the ETH Zurich by Pascal Müller (Parish and Muller, 2001) (Müller et al., 2006) (Wonka, 2006) and subsequently launched by Procedural in 2008 in Switzerland. Its primary target audience is the film and video games industries, focusing on the visualisation aspects of rich realistic cityscapes, but its generative features have attracted the interest of urban designers and features have been added to extend its use in this field. The generative features provide a programming interface for developing procedural grammars. However, the basic underlying structure for grid generation is not accessible for programming extension features at this level. The programming features address the development of grammars for designing building envelopes with particularly accurate detail on the design of façades. Although CityEngine was recently bought by ESRI there have been no major changes so far, at least in terms of integration with GIS software.
CityZoom is a Microsoft Windows tool developed over 15 years at the SIMMLAB of the Universidade Federal de Rio Grande do Sul (UFRGS) in Brazil by Benamy Turkienicz and Pablo Grazziotin (Turkienicz, Gonçalves, and Grazziotin, 2008) (Grazziotin et al., 2004) as a product of academic research. It is not available as a commercial package but has been presented at various international conferences and workshops. It is presented as a design support platform that complements the use of CAD and GIS systems in some specific urban design tasks, in particular simulations based on urban design regulations and the sunlight exposure interface. The tool does not provide a programming interface for developing extension features.
AutoCAD Civil 3D is a Microsoft Windows tool based on the popular CAD package from Autodesk, but extending it primarily with features specific to road and urban site design and several GIS features, in an attempt to merge both types of software package. This tool is freely available for academic use from the Autodesk website. Although the extensive amount of tools in this package makes this software one of the most complete commercial CAD tools, the CAD and GIS features are not integrated – they simply share the same working environment. However, AutoCAD has several programming interfaces using different programming languages which allow for great flexibility in terms of tool extendibility. The software package is provided with application programming interfaces (APIs) for two scripting languages – VBA and VLisp – allowing for tool extendibility.

The shortlist of software was then reviewed in the light of certain predefined criteria developed by the research team. The criteria for reviewing the shortlist of candidate software were defined by taking the three City Induction models into consideration (Gil et al., 2010) and considering what a CIM (city information model) should be. The software should:

1 Incorporate or allow for the incorporation of an urban ontology;
2 Respond to the planning regulations and strate¬gies defined for the site;
3 Consider the context, holding information on both the site and population;
4 Support the formulation of programs for urban intervention;
5 Provide for the development of design patterns for designing the urban environment;
6 Include a generative design model – a plug-in extension based on the application of shape grammars or other rule-based generative systems – or a shape grammar interpreter;
7 Analyse sustainability indicators;
8 Allow for interaction between data and design;
9 Provide an interactive visualisation of data;
10 Evaluate and rate different designs.
11 Include one or more application programming interfaces (API) for tool development or extendibility.

Table 12 shows a comparison of the four platforms in terms of these criteria. There was no platform that included all the criteria we needed to make an objective assessment of the candidate software and it became almost crucial that the final choice should be based on the existence of APIs for tool extendibility. CityCAD, although possibly the most complete software, offers no means of creating new extensions. CityZoom is limited, both as a design tool and an analytical tool and lacks a GIS connection. CityEngine provides an interface for programming grammars for generating building envelopes. This interface enables extensive realistic details of façades to be generated but has major limitations in terms of higher levels of decision such as launching the main planning guidelines e.g. compositional axes as structural streets and squares. For this purpose, however, it can be used together with other CAD software such as AutoCAD. The problem with this platform is that the most structural and high level decisions in urban design are not controllable in this software and no programming tools are available at this level. AutoCAD Civil 3D has very good CAD and GIS platforms which are not really integrated in terms of CAD-GIS interoperability but both environments are quite generous in terms of providing programming environments for extending tool capabilities and customising their use. A paper written by the City Induction research team addresses this survey (Gil et al., 2010) and concludes by selecting AutoCAD Civil 3D as the platform most suitable for the common goals of the research. This subsection was based on this paper.

Feature	CityCAD	CityEngine	CityZoom	AutoCAD Civil 3D
Incorporates an urban ontology	A	A	A	C
Responds to the planning regulations and strategies defined for the site	A	B	A	D
Considers the context, holding information on both site and population	C	B	D	D
Supports the formulation of programs for urban intervention	C	-	B	D
Provides for the development of urban design patterns	B	D	-	B
Includes a generative design model	C	B	C	D
Analyses sustainability indicators	A	-	C	D
Allows for interaction between data and design	C	C	C	C
Provides interactive visualisation of data	B	-	B	B
Evaluates and rates different designs	-	-	-	-
Includes one or more APIs for tool development or extendibility	-	yes	-	Yes

Table 12
Features required for a city information model offered by each of the selected urban design tools, indicating whether a feature is A – built-in; B – built-in and customisable; C – partially implemented; D – not implemented but extendable.

However, the detailed view of the generation module which is the subject of this thesis implies a detailed analysis of other tool functionalities regarded as necessary for automating the design generation.

§ 7.1.2 Survey of shape grammar interpreters

Shape grammar theory offers great potential for developing design systems capable of generating designs within some predefined formal world and exploring variations within this design world. The advantages of using shape grammars in design lie in the possibility of exploring and developing design languages from which a designer can select alternatives. The way a grammar is designed is analogous to any regular design process (Knight, 1999). The task of computing shapes, rules and grammars is reserved for the calculating capacities of the computer, leaving the designers free to consider design issues and the criteria for alternative shape, rule and design development or

selection. However, the implementation of shape grammars usually has to confront a paradigm: either a shape grammar implementation is started from scratch (1) or the shape grammar theory is adapted in order to implement it using the existing CAD software, taking advantage of the existing tools to avoid reinventing the wheel (2). The first approach allows for the correct orthodox implementation of shape grammars but it is not an easy task. In Section § 4.4 it was noted that *subshape recognition* is one of the main problems in shape grammar implementation. In order to deal with this problem, shape grammars are provided with the concept of maximal lines, which uses a particular algorithm to interpret two partially superposed lines as one line (Krishnamurti, 1992a). Conversely, when two lines cross each other they can be seen as four lines with a common starting point. The maximal line representation algorithm allows the user to work with shapes in very much the same way as they would do visually. Krishnamurti developed an extensive set of definitions for shape recognition dealing with maximal shapes, including algebras for lines, planes and 3D shapes (Krishnamurti, 1992b) (Krishnamurti, 1992a) (Krishnamurti and Earl, 1992). Stouffs (1994) summarises all the mathematical definitions involved in shape recognition for shape algebras in 0 to 3 dimensions. In any case, even given that the theory already solves many of the technical problems of subshape recognition mathematically, there are still problems regarding semantics. The recognition of a particular subshape in a design is not usually the main reason for applying a transformation to that subshape, but rather the meaning of that subshape in the context of the design.

Nevertheless, the problem is that there is no CAD software based on this paradigm except for a few limited standalone applications and a few very limited plug-ins. There are some common limitations to shape grammar interpreters. Many of them only compute 2D shapes. In most cases it is not possible to return to a particular stage in the recursion to change the rules and the recursion needs to be restarted. This makes the software very unfriendly for design purposes. In addition, most do not apply subshape recognition and therefore it becomes impossible to compute their emergence. Furthermore, in a creative process a design emerges when the recognition of a shape involving a particular meaning associated with its partial components emerges from a particular set of design occurrences which might not be easy to predict. Therefore, in a real shape grammar implementation, the emergence and the semantics of the emerging form still have to be controlled. What is claimed to be the best feature of shape grammars can in fact become a major problem because of the need to apply some judgement process to the qualities of what emerges. This creates a paradox: the proclaimed virtue of the emerging shape is the fact that we cannot understand its inherent qualities from the qualities of its components, as their composed qualities transcend their simple summation and therefore can only be recognised after a particular composition provides the emerging shape and the semantic evidence of it. Therefore, no judgement criteria can be defined in advance.

The second approach to shape grammar implementation has to confront the lack of any CAD software using the maximal line representation. However, given the difficulties involved in subshape recognition, adapting a shape grammar

implementation to the constraints of existing software is surely an easier option. This allows the implementation to make use of the existing drawing capabilities of the CAD software, although one can start from an existing shape grammar plug-in or from scratch on an existing CAD software. Furthermore, whatever the final choice, the platform needs to communicate as easily as possible with a geographical information system (GIS). Another important issue involved in evaluating the feasibility of a shape grammar interpreter is its ability to respect an acceptable design workflow.
Regarding urban design, a 2D shape grammar interpreter was never seen as a constraint on the implementation of the generation module since large scale urban operations are essentially planned on a flat surface or using flattened topographical height information. In addition, dealing with this kind of topographical issue has already been addressed in GIS systems research, in which most information is also flattened.
Gips (1999) made a detailed survey of the shape grammar interpreters developed up to 1998. Within the survey, Krishnamurti's pioneering work on implementing the first algorithms for 2D subshape recognition and rule application should be highlighted (Krishnamurti, 1981). Heisserman (1991) developed the first implementations of 3D shape grammars, using the topological structure of shapes to support subshape recognition. The main idea is that shapes are bounded by lower dimensional shapes; lines, for instance, are bounded by points, planes are bounded by lines and so on. These part relations are explored to find a subshape in a shape. In order to solve the technical problem of 3D subshape recognition, Stouffs' (1994) pioneering work developed a complete algebra for calculating with 3D shapes. Unfortunately, no complete implementation was produced from this work. Furthermore, in each case the shape grammar implementations are very limited in terms of their functionalities and usability.
After Gips' paper some other shape grammar implementations were developed. In this survey on shape grammar interpreters a few other recent implementations were considered and tested. Table 13 shows a list of these implementations and some of their characteristics.
The Yazar and Çolakoglu (2007) QShaper, a plug-in for 3DStudioMax, was the most designer-friendly shape grammar tool used and one of the first to be tested in this survey. The tests showed that 3D shape grammars could easily be implemented and reasonably complex 3D designs developed. As a test, a 3D implementation of a previously developed evolutionary housing system was used (see Figure 31, Figure 32 and Figure 33). However, QShaper was soon abandoned due to City Induction's requirement to select a CAD-GIS friendly platform, which was not the case with 3DStudioMax. It is, nevertheless, a very interesting shape grammar tool because it makes use of the modelling tools provided by 3DStudioMax.
With the exception of CityEngine, the other shape grammar interpreters in Table 13 were implemented in AutoCAD or the generations were exported directly to AutoCAD. Therefore they could all gain from the fact that AutoCAD already provides a CAD-GIS environment in the AutoCADCivil3D platform, although with limited interoperability.

However, SGTools (Romão, 2005) is the only shape grammar interpreter built in the AutoCAD environment that makes use of the tools already provided by AutoCAD. This shape grammar interpreter provides a new set of toolbars containing tools for defining 2D and 3D shape rules and tools for applying the defined rules. The software provides a virtual space defined in a hidden layer used as a reference for applying the rule transformations. The interesting thing about this shape grammar interpreter is that it allows the designer to use it according to his/her needs inside the CAD environment as an extra set of tools. Hence, the shape grammar functionalities extend the AutoCAD functionalities, maintaining the regular workflow of the software. However, once again the implementation does not allow for subshape recognition.

Name	Reference	Tool(s)	Subshape	2D/3D
QShaper	(Yazar and Çolakoğlu, 2007)		No	3D
City Engine	(Parish and Muller, 2001)	L-Systems	No	3D
SGI	(Li and Kuen, 2004)	Macromedia Flash	Yes	2D/3D
Shaper 2D	(MCGILL, 2004)	JAVA	No	2D
SGTools	(Romão, 2005)	AutoLisp	No	2D/3D
SGI	(Trescak, Esteva, and Rodriguez, 2009)	JAVA	Yes	2D

Table 13
Comparison of recent shape grammar interpreters or grammar-based interpreters.

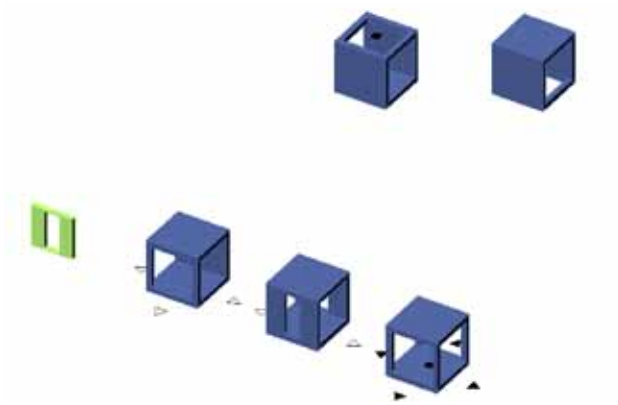


Figure 31
Test implementation of a shape grammar using Qshaper: Vocabulary

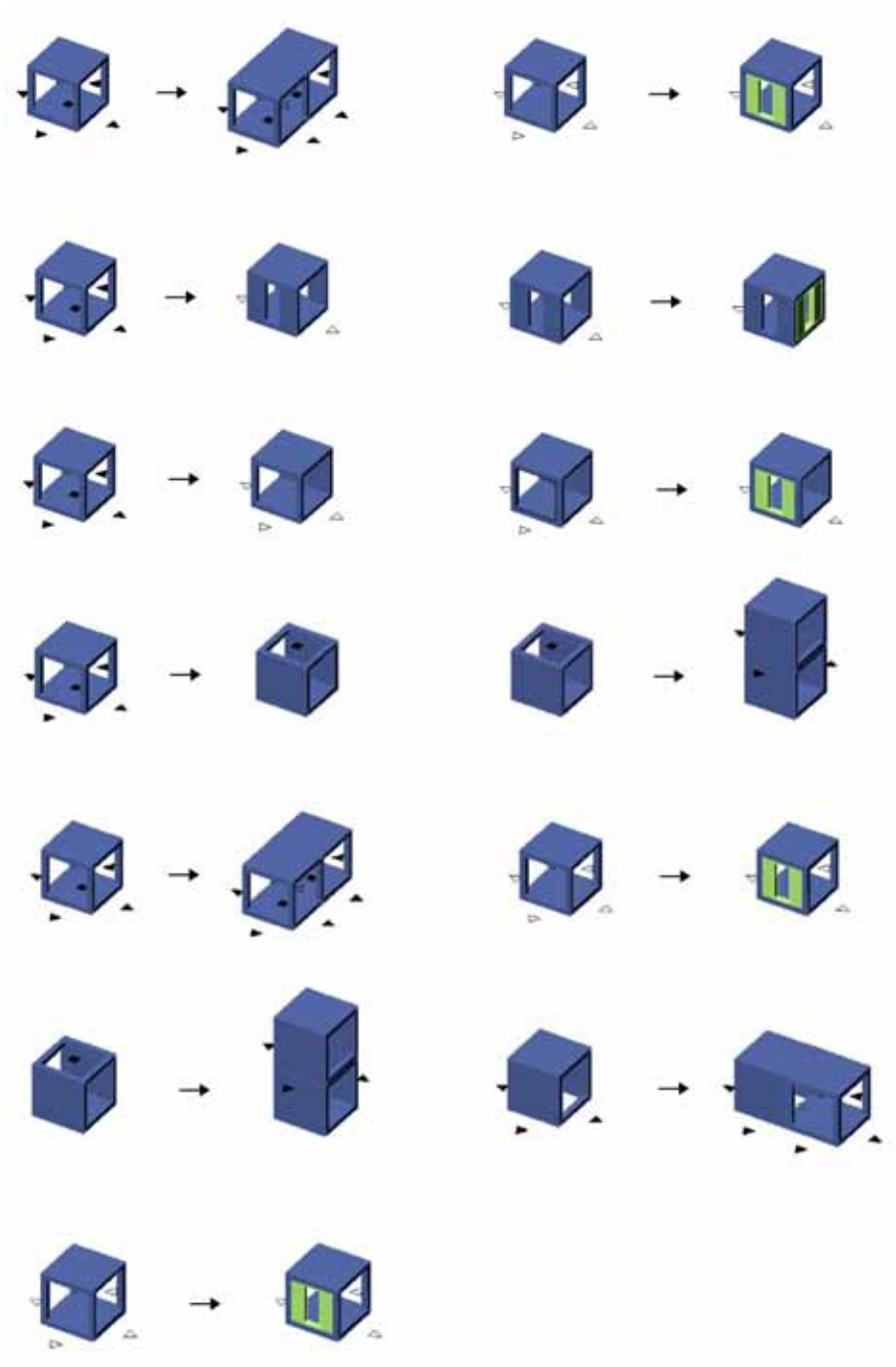


Figure 32
Test implementation of a shape grammar using Qshaper: Rules

The SGIs in Table 13 provide very correct interpreters in the sense that they follow the formal concepts of shape grammars quite accurately with regard to subshape recognition as the basis for rule application. However, the interfaces are limited in terms of usability and difficult to integrate with other CAD tools or with a GIS platform. Trescak's interpreter, for instance, provides the most accurate implementation of shape grammars including subshape detection, but only exports image files. Furthermore, it is very difficult to integrate images into shape rules within a CAD environment. These interpreters may be considered very interesting for developing certain abstract grammars but need to confront the difficulties involving communication with common CAD software.

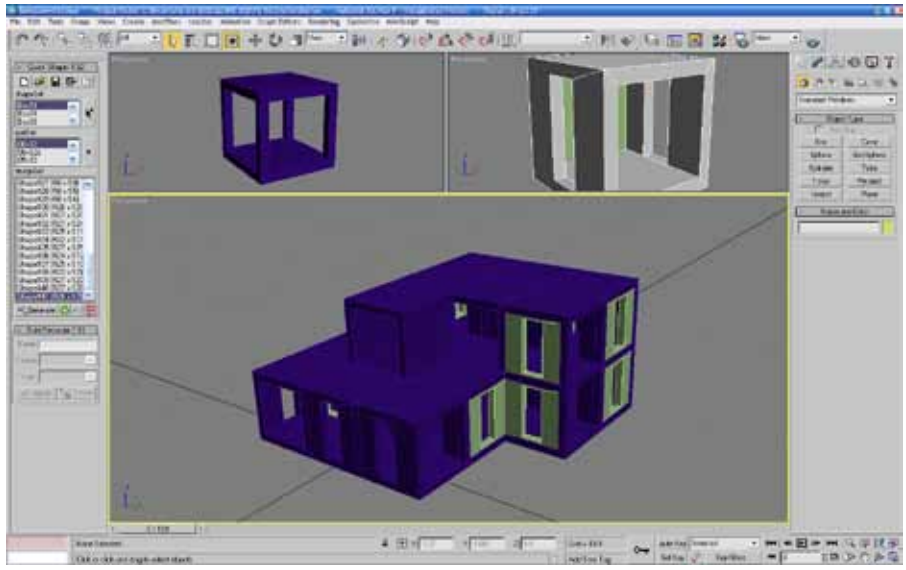


Figure 33
Test implementation of a shape grammar using Qshaper: Design

The generative algorithm in CityEngine is not exactly a shape grammar. The generative behaviour is based on L-Systems (Prusinkiewicz and Lindenmayer, 1991) as explained in their seminal paper (Parish and Muller, 2001). The term procedural grammar, used in some of their papers and website, better explains the concept which defines a tree of procedures by applying some customisable parametric rules to generate an extensive amount of building envelope designs starting from similar initial shapes. CityEngine has also a predefined set of procedures: first the generation of a grid from a set of input parameters, secondly the street profiles and the urban blocks using specific algorithms, thirdly the subdivision of blocks into plots, and finally the generation of realistic building envelopes. Only the latter allows for full customisation and rule development. The main parameter for initialising generation is the number of street segments, which is hardly a significant parameter for urban designers. However this procedural structure can be easily mapped into a data tree. Specifically, in CityEngine the descriptions of cities are clearly structured in a tree of components: the first level generates the network, the second level generates streets on one branch and blocks on another, and the third level details these two branches. The detailing part is similarly structured in a tree-like mode. The important thing to bear in mind here is that the complex structure of object classes seen in the ontology in Figure 9 is the one that urban designers manipulate while designing, rather than façades. Façades are actually one single object class occurring in the end branches of the ontology. The problem with CityEngine is that the first features are not detailed enough, nor are they available for manipulation.

Very few shape grammar interpreters allow for the implementation of rules operating with symbols, which were always envisaged as necessary to define design moves based on the semantic properties of components of the urban environment. The most sophisticated interfaces allow for the use of labelled points to define spatial relations. In any case, still a geometrical approach.

The main conclusion from this long survey of shape grammar interpreters is that it seems to be very difficult to develop shape grammar interpreters that respect the formal concepts of shape grammars. Additionally, the capacity of a shape grammar interpreter to compute emerging shapes plays an unclear role in design. Stiny (2005) and Knight (2003) say that shape grammar emergence emulates the designer's creative behaviour. However, before embedding emergent behaviour in a shape grammar interpreter a question should be asked: who needs to be creative – the tool or the user of the tool? If the answer is the user then emergence just becomes a disturbance within the overall process. Some people may argue that a system working with emergence may suggest unpredictable solutions, but this is likely to occur randomly unless some system of values immediately evaluates the emerging shape. This is problematic since, according to Lawson's design model (2006), the designer's perception of a design problem evolves along with the evolution of the design solution and this means that the system of values on which the designer bases his/her judgements also evolves throughout the design process. Donald Schön stresses the same behaviour when he refers to the designer's ways of seeing and how seeing changes from one move to another, affected by the move itself. Therefore, if the system of values evolves simultaneously with the design it becomes

difficult to develop efficient systems to evaluate the semantic qualities of the emerging shape. The whole problem regarding the use of emergence in shape grammars simply concerns the distinction between *developing tools for designers* and *developing tools for designing*. The latter aims at replacing the designer by emulating the creative act[26]. Although, shape grammars in their formal definition constitute the foundations of what might be a creative design machine, we still do not have enough knowledge to solve all the semantic details involved in creative reasoning. Finally, it is not the purpose of this research to do so.

This conclusion leaves us with shape grammar interpreters which still rely on designer interaction to make creative decisions. In this case, three typical situations occur. The first, and most common, involves a standalone interface with limited functionalities that is difficult to integrate into other CAD environments and does not integrate at all with GIS environments. The second situation presents shape grammar interpreters with an interface embedded in an existing CAD platform. These are definitely the most designer-friendly shape grammar environments of all the shape grammar interpreters cited. SGTools is the only shape grammar interpreter which is integrated into a CAD software package that already has basic GIS interoperability – AutoCAD Civil3D. SGTools is also the only shape grammar interpreter which already provides for extending development of the same application programming interface that allows for its implementation. The third situation is CityEngine, a standalone autonomous software package with a procedural structure that implies a predefined method of use. The main limitation however, is the scarce means for controlling the first levels of the urban design process.

To avoid the limitations resulting from subshape recognition and semantic problems resulting from the ambiguities of emergence, the best option was to adapt the grammars developed in the research to the characteristics of the chosen software platforms used to implement Models A and B. In this way it was possible to use the powerful interfaces of the selected CAD platforms and respective APIs and take advantage of all the existing CAD features without the need to set up a CAD system from scratch. In both Models, the generic urban grammar developed as foundation of CItyMaker's theoretical framework was adapted to meet the available features of the CAD platforms. The adaptation of Model A is nevertheless very accurate and although it does not support emergence, it follows the theoretical model in detail.

26 The nature of the creative act is still open to discussion. Taylor (1959) identifies five levels of creativity, the lower ones essentially identifying decisions at the level of productivity involving progressive steps in proficiency, and the higher levels of creativity, which he calls innovative creativity and emergent creativity, in which the former involves innovative transformations of the assumptions underlying the subject in question and the latter the creation of new paradigms involving changing the assumption that is the basis of some work or knowledge field. The higher level of creativity in particular implies an unpredictable change in the system of values underlying the subject of the creative act. To complicate things further, if Duchamp's idea of the role of the spectator in forming the value of the artistic creation is taken into consideration (1957), then the possibility of predefining the production of a creative act becomes impossible using any available method or discipline. However, it does not seem correct to extend Duchamp's idea of the creative act to urban design. The parallel is too imprecise.

§ 7.2 Model A – Implementation in AutoCad using VBA and VLisp APIs

Moving from theory to practice, a computational prototype was developed using AutoCAD Civil3D. As explained in the previous section, this platform was selected by agreement by the City Induction researchers.
AutoCADCivil3D was chosen because it combines the CAD features provided by the regular AutoCAD drawing and modelling tools with the GIS features provided by AutoCAD Map, along with the terrain modelling tools available in Civil3D. Representations can be accessed in these three workspaces as long as the designs generated are exported to the Map platform with the appropriate object representations following regular geo-referencing procedures. Designs and data can be accessed from within the GIS module. The evaluation routines can use most of the available GIS-based assessment tools during the evaluation procedures, as well as other tools developed specifically for this purpose.
Model A was implemented using the AutoCAD Integrated Development Environment (IDE). Visual Basic for Application (VBA) and Visual LISP (VL) were the programming languages used. It was possible to use the AutoCAD Application Programming Interface (API) through the VBAIDE and VLIDE, as well as the graphic editor and the object structure already available in the tool, thereby facilitating the implementation process for the proof of concept prototype.
The AutoCAD object model contains pre-defined methods and attributes, object classes that have inherited and extended functionalities from the standard API. This is similar to the structure of an object oriented programming language and the object structure is compatible with an ontology structure such as the one developed for the urban design process as shown in Section § 5.4 (see Figure 9). This means that object classes can be stored in the City Induction ontology. However, to maintain the autonomy of this prototype as an independent design tool the main features of this model were implemented with the VBA API, creating the object classes from scratch and making use of the object oriented structure of this environment. In using this process, the main concern was to associate metadata with the objects being generated, allowing the prototype to work with a semantic structure rather than just geometric and syntactic structures. For instance, the street axes representations use attached external data – a classification identifying the street hierarchy – which can be read by a rule predicate to apply a consequent transformation based on the street classification rather than its formal features[27]. Using these abilities, the elements generated in the

27 This can be accomplished by using data links or *XData* in AutoCAD. Data links store data in files – excel files or access files, for instance. *XData* stores a limited amount of external data in the design entity itself. This data can be extracted at any time from the entity selection using the *List XData* command. Both strategies were used in the implementation of Model A.

design contain information about themselves and their relative position in the semantic data structure such as which object class they belong to. The object class structure developed allows for accurate correct communication between the CAD and GIS environments by establishing the direct translation of objects in the CAD structure into objects in the GIS structure. The most important detail is to guarantee that the objects generated are arranged in separate layers depending both on geometry type (point, line or polygon) and object class as defined in the ontology. Therefore, every class has, in principle, at least three separate layers, although thematic variations on this basic structure may be contemplated, for which new layers may be developed.
Attaching metadata (*XData* or *external data*) to the AutoCAD objects allows for the possibility of using such data to control various semantic aspects of the design, for instance, by predefining parametric relations involving not just geometry but also the semantic qualities of the objects. In addition, by linking object properties to data on a spreadsheet, it becomes possible to extend the control and use of external data during the design process. The use of *XData* and *DataLinks*[28] in AutoCAD allows for two important aspects: firstly, it makes this the mechanism used in this environment to deal with the concepts used by the description grammars and, secondly, it is also the mechanism that allows information useful to the design process to be stored, such as calculations of density indicators.
The implementation followed a well-known method in computer science called incremental development, which consists of starting with very simple routines and then progressively adding new expressions, variables, controls and functions to obtain complex programmes. The VBA implementation was developed with the help of VBA programmer Gelly Rodrigues and supervised by the author of this thesis.
So far, patterns have been developed to generate the compositional axes of the urban plan, to generate grids, and to generate urban blocks. A list of some of these patterns is shown in Table 14. The icons found in Table 6 correspond to new tool buttons from a set of new toolbars that were added to the AutoCAD workspace. The complete set of new toolbars corresponds to the CItyMaker prototype Model A (see Figure 34).
However, only a few of the tools have already been implemented in the model and they correspond to the ones in bold type in Table 14.
Patterns can only be applied if certain features exist in the workspace. If such features do exist, the rules take them as their initial shapes or symbols and trigger the rules that are applied recursively until an end state occurs. The end states are defined within each pattern as a set of conditions that always occur within the application scope of the pattern. A UIP can create new elements in the design that might be read as initial features by other patterns activating new UIPs for application. Patterns are selected

28 *XData* stores a limited amount of data in the AutoCAD entity (e.g. a line with a number, class identification, a name – for instance, Main Street – and a street hierarchy – for instance, a_2). *Datalinks* attach data to a file, for instance a spreadsheet or an excel file, allowing it to store and relate any information that may be associated with the entities in the plan (e.g. all islands can be stored in a table containing entname, block type classification, area, street type facing the sides of the block, density, etc).

from the available active UIPs. The common initial elements in any design derivation basically consist of two types: (1) the intervention site, an *AcadPolyline* entity, and (2) a set of urban design references, which are existing line or point entities selected by the designer from the existing environment to inform subsequent design decisions by the UIPs. These references are *AcadEntitys*, to which the attribute R_{ef} is added so that they can be recognised and used by the first patterns.
This section shows the application, using Model A, of the following small set of patterns: *MainAxis*, *OrthogonalAxis*, *AddingAxes*, a grid generation pattern and *AddBlocktoCells*, which inserts islands in the cells generated by the grid of axes resulting from the application of *AddingAxes*. The application of *BlockType* to replace islands by specific block types is also shown but it corresponds to an independent application developed with the VLisp API.

§ 7.2.1 Preliminary implementation results

The first step in the design process is provided by the programme formulation module and consists of an analysis of the design context, extracting and analysing the available information in order to produce a list of specifications - the urban programme - to be taken into account in generation. However, it might be difficult or virtually impossible to know all the inputs needed for the generation module in advance as some inputs might depend on the reaction to what is being generated. As such, the urban programme might be incomplete and might be continuously refined and updated. Considering this possibility, the design system is implemented in such way as to request data each time it is needed and not found in the available specifications. In the current prototype implementation, information is requested from the user (the designer) who should be able to fill in the gaps as needed. In addition, other information might be subjective and concern stakeholder expectations or idiosyncratic design options. The implementation is therefore open and interactive to accommodate such nuances.

Figure 34
New toolbars with the CityMaker tools.

The number of patterns available for use in the design depends on the progress of the design itself. The first patterns available for application react only to two kinds of initial shapes, the intervention site border (I_s) and elements selected by the designer as design references (R_{ef}). These references can, in principle, be any elements of the design context that may be selected by the designer as referential guidelines for the design, such as landmarks and focal points. The current version of the implementation considers only weighted points in the design context marked by the designer to identify such design references. By connecting these reference points the system provides directions that can be used to trace compositional axes, much in the way designers usually do.

When a pattern is applied, the result may turn existing patterns off and turn additional ones on. The available pool of patterns is thus constantly being updated. This behaviour is induced by the explicit relationships between object classes which are expressed in pattern predicate conditions and in indication of related patterns that is part of a pattern's definition (see Appendix 2). The ontology in Figure 9 is in fact the schematic representation of the relationships expressed between objects used by patterns in the generation model.

So far, only a few patterns have been implemented in Model A. The design options are mainly limited to exploring the parametric variations within each pattern. However, a larger set of patterns will extend the design options to encompass other morphological domains. At the moment, this is more perceptible at the level of block generation, where the differences between block types confer a different character on the grids generated.

Figure 36 shows the application of the following sequence of patterns: *Cardus* (an option of *MainAxis*) + *OrthogonalAxis* + *AddingAxes* (grid definition by adding axes) + *AddBlocktoCells* (adding islands in the cells defined by the grid, considering the street width stored in street representations).

Urban Induction Patterns ▼Case studies ►		CG – Praia	CG – Moit	FD, FC, TS – IJburg	P – Ypenburg	Cerdá – Barcelona
Creating the composition guidelines						
MainAxis	**MainAxistheLongerLine**		•	•	•	•
	Cardus	•				
	MIAxis (Most Important Axis)					
OrthogonalAxis	*OrthobyMidPoint*		•			
	Decumanus	•				
	MIAxis (Most Important Axis)	•	•	•	•	•
InsertFocalPoint		•	•			
CompositionAxis			•		•	•
Creating grids (completing the street network)						
(grid by) AddingAxes		•	•	•	•	•
(grid by) *AddingBlockCells*		•	•			•
Transformations to street network						
AxisOverGrid					•	•
MoveGridNode				•	•	•
Creating Public Space						
AddPlaza + GeneratePlaza		•	•			
InsertPublicSpace + Square	*Sq2*	•	•	•	•	•
	Sq3	•	•	•	•	•
	Sq4	•	•		•	•
	Sq5			•	•	•
Creating Urban Units						
AddBlocktoCells		•	•	•	•	•
AdjustingBlockCells		•	•	•	•	•
ClassifyUUnitCells		•	•	•	•	•
DefineUUnit	**BlockType**	•	•	•	•	•
	Cluster			•	•	
InitialUUnit				•	•	•
AddUUnitbyLabel	**AddBlockType**	•	•	•		•
	AddCluster			•	•	
InsertPublicBuilding (service buildings)		•	•		•	•
InsertBuilding (inserts a building in a focal point)		•	•			•
Others – detailing patterns and assigning functions						
AddAccessStreet					•	
AssignFunctions		•	•	•	•	•
ManageBuildingHeight		•	•	•	•	•

Table 14

Table of urban induction patterns and their application in the case studies. The patterns in bold are the ones that have already been implemented.

The design workflow starts with the selection of referential elements for the design (R_{ef}s). This is done by manual selection (or by drawing points) in the existing representation of the intervention site I_s and/or surrounding areas. Later the designer may specify the importance of the R_{ef} elements by attributing different weights to each of them. Weights are attributed from 1 to 5 and are stored in a database along with an identifier (in this model, points are defined as R_{ef}s) and the geometry of the design entities – Figure 36 and Figure 37. In brief, these urban induction patterns have the following generative behaviour:

MainAxis - calculates all the possible axes inside Is that can be traced between all the selected Ref points and chooses the following criteria: the LongerLine, following the maxim of author of the Ypenburg plan, Frits Palmboom, - "I always look for the longer line in the territory"; the *Cardus*, which chooses from the candidate lines the one closest to the north-south orientation; *MIAxis*, which chooses the most important axes by considering the weights attributed to the R_{ef} points. *LongerLine* and *Cardus* are objective options. There will be only one longest line and only one *cardus*. The *Cardus* is the line taken from the set of candidate axes that is closest to the south-north direction. The candidate axes are pre-selected considering two aspects of the lines generated: their length and their weight. The length is taken from the geometry and stored on a spreadsheet. The weights are the sum of the weights of the R_{ef}s that define the lines. The designer decides whether to place more relevance on the length of lines or their weight, thereby defining a relative percentage of incidence to apply to each criteria – length relevance ($l_{\%}$) and weight relevance ($w_{\%}$). This selection is made from a previous selection also controlled by the designer in which the smaller lines can be filtered out. The system always calculates the length of the longest line and the designer simply decides, based on a percentage of the longest line length, on the minimum length from which lines should be considered as candidate axes. The candidate axes are highlighted on the screen allowing for visualisation of the lines selected for the set of candidate axes. The lengths of the candidate lines are standardised to values between 10 and 2 – 10 for the longest line and 2 for the shortest one. The most important axis is calculated using the following equation: $M_{IAxis} = (A_{xLgth} \cdot l_{\%}) \cdot (A_{xWeight} \cdot w_{\%})$. The values for M_{IAxis} fall between 100 and 4. The algorithm chooses the line with the highest value as the main axis and attributes a label a_1 to it identifying the corresponding street hierarchy.

OrthogonalAxis - calculates all the possible axes that are perpendicular to a selected axis and pass through an Ref point and then selects one of two optional criteria: *OrthobyMidPoint*, which corresponds to the axis that passes through a point closest to the midpoint of the selected axis; *OrthobyLongerLine*, which corresponds to an algorithm similar to the one in LongerLine; and *OrthobyMIAxis*, following a similar algorithm to the one used in *MainAxis*. If the axis is perpendicular to a *Cardus* it will also be marked as a *Decumanus* and, like *Cardus*, can only be applied once.

AddingAxes - generates a grid of streets and blocks considering a user-defined block dimension. The implementation follows the rules for this pattern shown in Appendix 2 – UIP 009. All the axes generated contain data on their hierarchical level (out of 4 hierarchical levels, a_1, a_2, a_3 or a_4) to which a set of user-defined street widths will correspond. This programming information is inserted into the system in advance using a tool called Urban Parameters (see Figure 35).

Figure 39 shows the final result of the application of a sequence of patterns to generate block types. It starts with a predefined orthogonal grid composed of a random number of streets with a random distribution of four hierarchical street levels, expressed in terms of street width, in which block cell dimensions vary randomly within a predefined interval. This set of patterns was implemented separately using the VLisp API. The main ideas behind this partial implementation were twofold: (1) to prepare the tools for generating block types based on the principle that the procedures previously shown would have already designed a grid of orthogonal streets with islands; and (2) to investigate the potential of storing metadata in AutoCAD entities instead of storing it in a spreadsheet. The main idea behind the last point was that this kind of procedure would result in a faster programme, that is, reading data in the entities would be faster than reading it from a spreadsheet. For the same purpose, we would then need a grid of *islands* containing information on the hierarchy of the streets surrounding them as metadata. To this end, an independent algorithm defined as a simplified version of *AddingAxes* generates a grid with a random number of rows and columns (the user defines a range, e.g. between 8 and 12), with blocks of random widths and lengths within a user-defined interval and a random distribution of street hierarchies for which the user sets the corresponding widths in advance. The result is a set of *islands* like the ones found in Figure 38. Each island contains a list of the information needed to apply the following patterns: *ClassifyUUnitCells* – assigning different classifications to *islands*; *DefineUUnits*, option *BlockType* – defining a user-defined number of block types; and *AddUUnitbyLabel*, option *AddBlockType* - replacing the *islands* with the types that correspond to the predefined block classifications, adapting each type to the existing block dimensions (see Appendix 2, UIPs 027 and 030).
List processing allows any kind of information that may be represented in the form of a list or a list of lists to be stored. Metadata can also be stored in AutoCAD entities in the form of a list of attributes or values. As such, the *islands* in the randomly generated grid in Figure 38 contain a list indicating the hierarchies of the streets surrounding them, ordered by their cardinal points: south-facing street, west-facing street, north-facing street and east-facing street. This list is actually added to a list containing the *dxf* code for the *island*. This allows the VisualLisp code to access any characteristic of an *island* in the design, as well as the selective processing of *islands* and their respective characteristics.

The patterns referred above execute the following procedures:

ClassifyUUnitCells - the user defines how many different block types to assign to the *islands* in the grid and which types are to be generated in each island. This is done by user selection. The selection stores a string in the AutoCAD entities identifying the classification. In fact, it adds a string to the list already stored as metadata. The string is a block type classification. If four block types are supposed to be distributed, then the *islands* will be classified as B_A, B_B, B_C or B_D.

DefineUUnits is supposed to allow for the definition of urban types of different kinds (blocks, clusters of buildings and neighbourhoods) but *BlockType* is the only one that has been implemented so far. *BlockType* defines all the characteristics of a specific block type. The user defines the block by combining two primitive block types out of a set of five (*island*, *closed block*, *linear block*, *punctual block* and *matrix block*), by applying one of four available operations (subtraction, addition, product and symmetric difference) (Stouffs, 1994). In principle, the designer defines the same number of types as the number of type attributes assigned to the *islands*, although storing pre-defined types in a library is envisaged as a better practice because it allows types to be reused, thus saving the time spent recording their characteristics each time. A block type is defined in a default but customisable *island* (100m x 100m). The algorithm records the block type definitions (design sequence and parameters) in a list which will be called up by *AddBlockType* during the generation of the blocks in the grid.

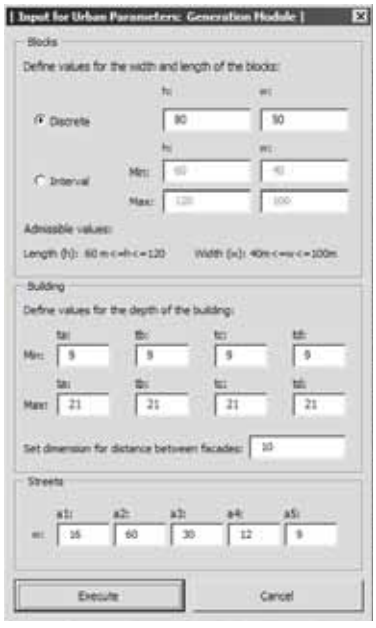


Figure 35
Input for Urban Parameters. a_5 is the street applied in the matrix primitive block.

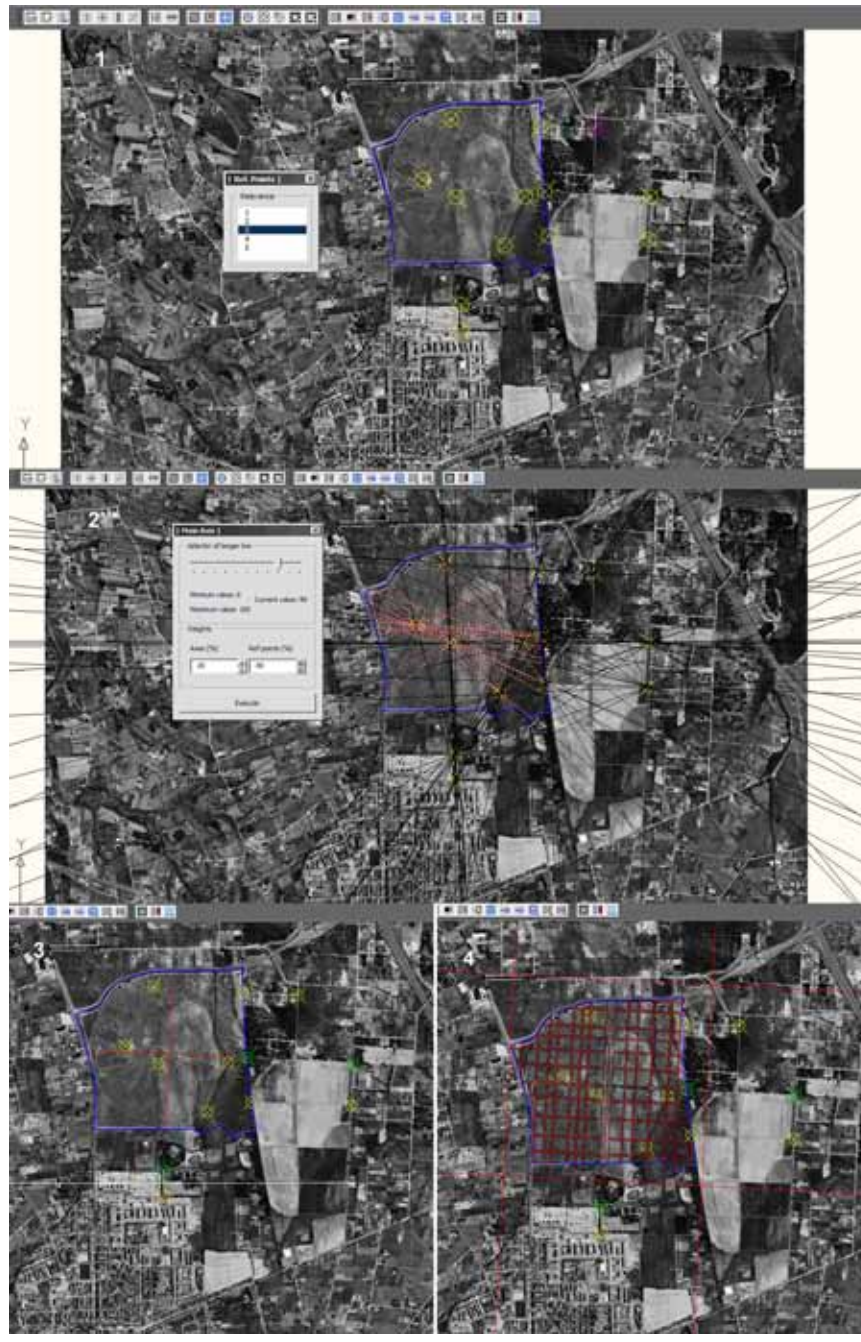


Figure 36

Grid generation: 1- Assigning weights to reference points; 2- Main axis selection interface; 3- MainAxis + OrthogonalAxis; 4- Grid by AddingAxes + AddBlocktoCells + AdjustingBlockCells.

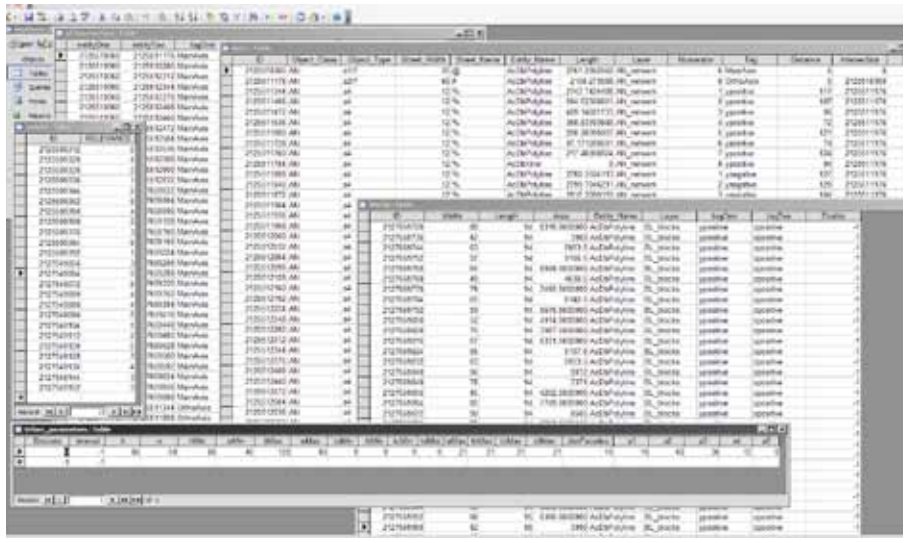


Figure 37
Generated data in the database.

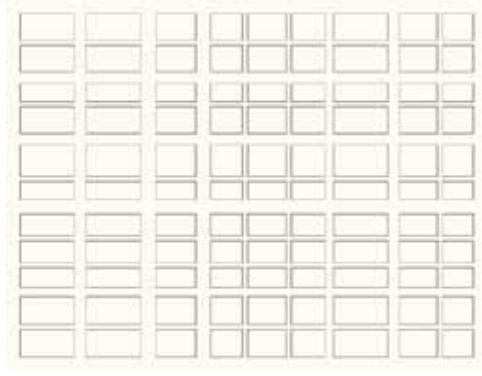


Figure 38
Grid of islands for the generation of block types.

AddBlockType replaces *the islands* in the grid with the corresponding block types, adapts the type to the island geometry and rotates it so that the front of the block faces the street with highest hierarchy. The pattern contains several routines that restrict the block structure and building sizes to reasonable and acceptable standards, for instance, constraining the bounding values for minimum and maximum building depth, minimum distance between the façades of separate buildings, etc. These values are set a priori in the design interface.

Figure 39
AddBlockType - this pattern replaces predefined block typologies in a grid of blocks orientating the front of the block to the main street. The 4 user-defined block types can be seen on the right of the image.

§ 7.2.2 Discussion and future work

The computerised implementation of Model A confirmed that urban induction patterns have a very complex structure, which constitutes a challenge. As observed in Alexander's pattern language, many patterns are embedded in other patterns and every pattern calls for other patterns. Similar situations occur in this implementation. For instance, *ClassifyUUnitCells* calls for the application of *BlockType* to define which block types to apply to the *islands* classified. During the design process, defining a specific distribution of buildings in a block presupposes specific plot subdivisions and vice-versa. Therefore, a pattern for generating buildings should also call up and constrain the pattern rules for generating plots and vice-versa. Although the previous development of urban space ontology helped clarify the relationships between the various patterns, the computerised implementation added greater awareness on how to relate and build up these patterns. It also helped providing a deeper awareness of the difficulties involved in automating relationships between patterns.
Developing an implementation is, in itself, a design act. In the case of a design tool, this means that we are confronted with different possibilities regarding which aspects to leave for user decisions and which to automate. An extensive list of possible improvements became clear during the work and the following are some examples of work planned for the future:

- Defining references to stimulate design options is a very important aspect of the design process and it is not only related to the position of the elements but also to their meaning and shape. Improvements include (1) creating linear or polygonal references (R_{ef}s assigned to lines or polygons) and (2) relating references to meanings (e.g. distinguishing references such as historical landmarks from natural features). Weights or priorities can also be assigned to meanings, for instance to define exclusions in certain contexts, such as excluding industry from historical sites.
- Developing some automated approaches to the classification of *islands* according to urban design data and criteria such as density, street hierarchy, and weighted references which, in turn, will require the development of a multi-criteria weighted approach[29]. The criteria for classification and the distribution of classification should be established with the support of several appropriate case studies.
- Automating plot subdivision following block design decisions.
- Storing predefined types for reuse or transformation in other design contexts. Storing could also be useful in terms of retaining information on the context in which block types have been applied so that they are made available only when such contexts occur. This feature could introduce some artificial intelligence into the design system, improving its capacity to make solutions fit contexts. These situations could easily be developed by storage including the block type definitions and formal descriptions of the contexts for which a block type application is allowed. These are easy features to implement within the existing structure.
- The grid generation needs to be improved in order to always obtain the correct topological structure for street segments and nodes. In the case of *AddingBlockCells*, despite the fact that the pattern is quite complicated in terms of rules, the structure of the rules actually facilitates this goal because streets are generated by segments. In the case of *AddingAxes*, additional rules are needed to capture all the segments resulting from street intersection[30]. The new rules could store all the street segments in the database, including street names and the coordinates of their end points, which would correspond to street nodes. This would provide a structure identical to GIS that would be available for design purposes. The main rules are already available and most of the information is already calculated and stored, but a few extra rules need to be

29 These two first points could also be achieved by thoroughly integrating this system with the ontology being developed by Montenegro et al (2011) – most of these goals may be easily accomplished by editing rules directly in the ontology. In this case, the classification of design elements would be automated using information directly associated with the design entities in use. However, it should always be possible to manipulate weights because this is a common approach in design practice and designers feel that they can manipulate this kind of expression with the tool.

30 This process was actually started, but has not yet been implemented in the main programme because it raised issues regarding the consistency of the whole programme due to the lack of a maximal line structure in the platform. However, these issues may be overcome by further work on the relations expressed between primitive representations defined in the ontology.

programmed. If this goal is achieved, we could explore the use of topology analysis methods in direct relation to design practice. The latter suggestion, however, is a vast research field in itself.

- Developing a user-friendly data output interface showing only a synthesis of the useful data and eliminating information unnecessary for design purposes. The idea would be to provide an interface just for the purpose of informing designers about density-based indicators.
- Implementing a larger number of UIPs. This should be done after fine-tuning some generic but more structural aspects of the system, such as some of those stated above. Such an extension would certainly provide a wider space for design exploration, enhancing the design capabilities of the tool and consequently its acceptance by designers.

Two points should be further discussed before ending this section.

1 The UIP formalism enhanced with customisable features such as those described in the example of block type generation seems quite a useful feature. In addition to the fact that it enables the system to be extended, it also allows for the customisation of UIPs, therefore improving its potential in terms of synthesising a design language. Judging from the previous experiences in design studios with shape grammars (see Section § 4.1) it is expected that this design tool will also enhance designers' awareness about their own language convictions. The possibility of storing the customised UIPs is a very interesting issue for further exploration and is easy to implement. Firstly, the design tool with this feature is continuously being extended, thereby extending the design space to personal fields of design exploration. Secondly, every UIP could benefit from having customisable features. This means that designers would be able to customise every design move, not just through parametric variations but also by inserting features of their own creation. This approach needs further research, but should be considered an important goal for this particular implementation.

2 The separation of block type generation from the main design model can also be justified from a procedural point of view, since the design of the plan guidelines, grids and blocks all correspond to different design phases. In most countries, regardless of the particular legal and regulatory framework, most planning procedures evolve through these distinct phases which are, or at least should be, submitted for (participatory) approval. As such, processes which are considered part of the same tool might be applied at different moments. In this sense, there is no objective need for all the tools to be fully integrated, as the output of one phase can be read as input in the following phase. Nevertheless, integrating all the tools offers the possibility of deciding when to stop and it can be a useful strategy for showing stakeholders some possible future scenarios before they approve a particular phase.

§ 7.3 Model B – Implementation in Rhinoceros using Grasshopper

Model B was developed with Pirouz Nourian starting from the main theoretical framework developed in this thesis but defined using a parametric design platform and adapted to fully explore the possibilities of the said platform. The tool was deliberately created for design at district level and/or neighbourhood design. Later on, further developments of this tool were elaborated with the collaboration of Pedro Arrobas[31].

§ 7.3.1 Translating a shape grammar system into a parametric system

Supporters of shape grammars will probably say that shape grammars embed almost all the known forms of generative design systems. They embed L-systems, fractals in general, cellular automata, and even parametric shape grammars can be said to embed a parametric design system (PDS). Theoretically, this is true, but the limitations on the implementation of shape grammar interpreters leave us with the unavoidable evidence that some specialised types of software such as parametric design software produces better and faster results than correctly implemented shape grammars. Additionally, PDS in general presents very user-friendly design environments which also explain the great success of PDSs in recent years, particularly Grasshopper which has recently caught the attention of most of the generative and parametric design communities. A simple search on the internet will reveal a large number of designers fully involved in these practices.
During the research developed for this thesis and before the start of Model B, some investigation had already been undertaken into parametric design using Generative Components, a Bentley software [WS11], but, despite the interesting results, the system seemed too heavy to deal with large amounts of information and large plans. Figure 40 illustrates some of the results of this research. Later however, the Rhino – Grasshopper software offered much better potential and performance than Generative Components. Nourian's suggestion of implementing CItyMaker as a PDS on Grasshopper was welcomed and work began in late 2010.
Nevertheless, the shape grammar model had to be translated into a PDS. Shape grammar systems are defined by sets of shape transformation rules applied recursively from an initial shape to generate designs. These systems produce a family of solutions by recursively applying formal composition rules which simultaneously guarantee

31 The results of these later developments have not yet been published. The model described in this thesis is mainly the outcome of the work developed with Pirouz Nourian. Recent results may be found at [WS12].

formal coherence and diversity. In order to obtain semantic design coherence, shape grammars can be combined with description grammars and heuristics, as proposed by Duarte (2001) and also proposed in this thesis for the application of urban design systems. However, semantic coherence can only be achieved if the shapes and concepts used by the grammars are correctly structured into an ontology, as defined in Section § 5.4 , from page 92 onwards.

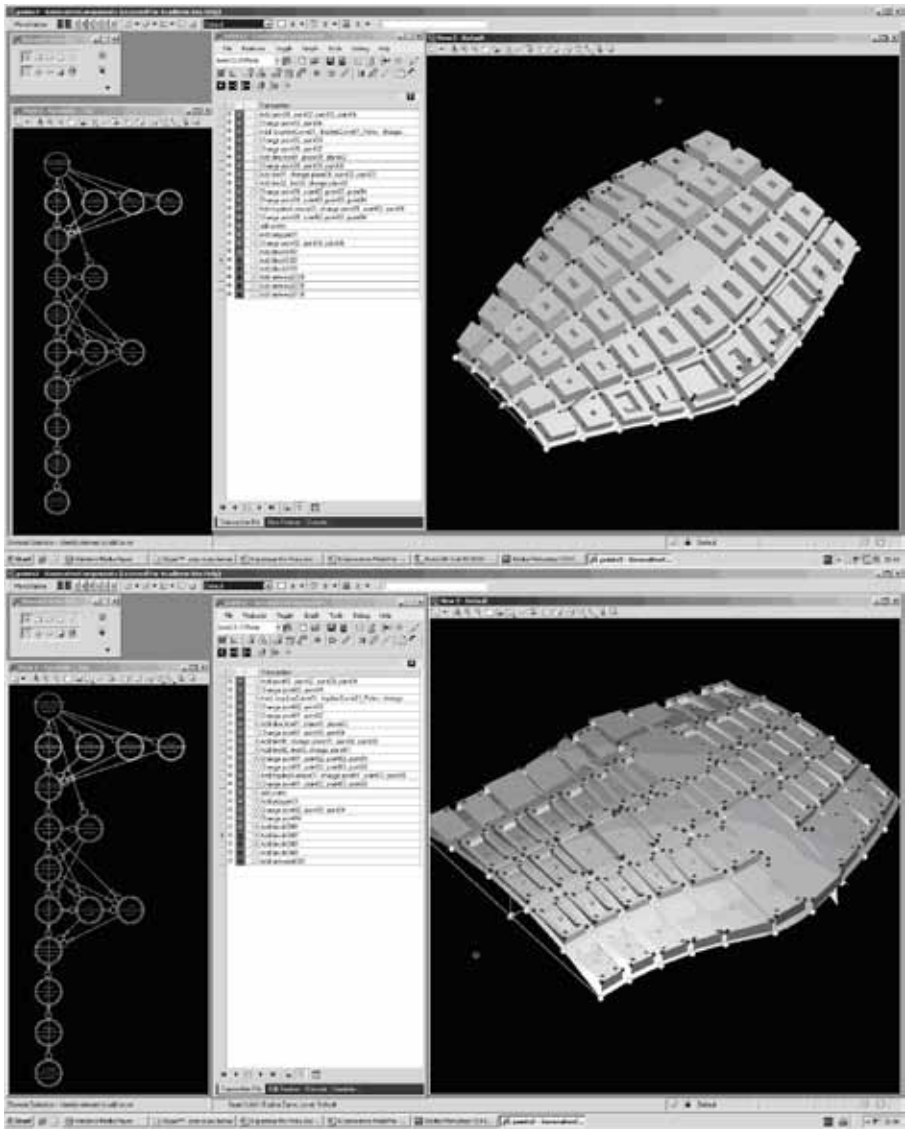


Figure 40
Trial implementation of Generative Components. The image shows two different results.

A parametric design system consists of a geometric model of formal elements which are common to a set of designs. The parametric model is built on the topological relations between formal design elements and their dimensional variations, expressed through a set of parameters assuming values within predefined ranges. Contrary to shape grammars, a PDS produces only one design solution with a large range of parametric variability.

In order to translate the shape grammar model into a PDS whilst maintaining the possibility of defining a coherent and diverse set of solutions, the implementation of Model B, like the theoretical model, followed a design pattern structure slightly adapted to the characteristics of the Rhino – Grasshopper platform.

In the theoretical model UIPs were arranged in thematic sets, as can be seen in Table 6. There is a set of UIPs (A) used for defining the composition guidelines of a plan, basically generating compositional axes. In addition, set B generates grids, set C transforms axes, set D generates public spaces, set E creates urban units, isolated exceptional buildings and in general manages the distribution of the built environment, and set F manages details of the plan with a particular focus on street design including the design of junctions. It is also stated in the theoretical model that a set of initial elements needs to be used to initiate the generation process, mainly the intervention site area and/or urbanizable areas represented by polygons, and the referential elements represented mainly by points, but which can in fact be represented by points, lines or polygons according to the design context.

Therefore, after developing the first test models, the implementation of Model B has been progressively refined into a design pattern structure (Woodbury, 2010) following the structure defined in theoretical model, namely the main concepts defined in the structure of urban induction patterns. In order to obtain the maximum possible adaptability for different design problems and contexts, the PDS was structured as follows:

1. A set of patterns to deal with the initial features – intervention site polygons and pre-existing elements defined by polygons representing sites, existing buildings represented by polygons, and existing streets or roads defined by lines.
2. A set of patterns to define composition guidelines – lines representing new streets.
3. A set of patterns to define grids – rectangular, radial and recursive grids[32] – applied to one or multiple polygons.
4. A set of patterns to define public spaces – filtered from a grid.
5. A set of patterns to define exceptional buildings or public buildings – filtered from a grid.

32 The concept of the recursive grid will become apparent in the description of Model B. Appendix 2 shows UIPs, *RectangleDissection* and *IcerayGrid*, which are two examples of a recursive grid. Other types of grid patterns were planned in some annotation sketches but have not yet been implemented. The recursive grid pattern corresponds exactly to *RectangleDissection*.

All these patterns use input geometries taken either from existing elements or proposed geometries drawn by the designer. An additional set of patterns manages other type of design input, namely:

6 A set of patterns to manage density – including the input parameters for a desired maximum building height, and a desired value for minimum private open space (minimum OSR at island level).

7 A set of patterns to manage land use distribution defined in terms of the distribution of residential uses, commerce, public facilities, workspaces and small industries.

Other parts of the system are also defined as design patterns, since they are also repeated:

8 Patterns calculating density indicators.

9 Patterns filtering parts of a plan. The filtering process allows different design inputs to be applied to the filtered elements. The filtering process works by filtering an area by selection, placing a point or using a polygon, and filtering by proximity or by intersection.

10 Patterns for data visualisation (e.g. pie charts, block indicators and colour gradients for blocks – see Figure 43).

11 Patterns defining attraction fields for design elements. These patterns may be called attractors and are the main components of the patterns mentioned in Points 6 and 7.

The following parts of this section will describe Model B in detail, providing additional insight into the subject and showing how the structure was implemented. However, it should be noted that the system is still being improved, both in terms of the adaptability of the pattern structure and the development of new additional features (for instance, the implementation of patterns for designing street profiles following the structure shown in Table 5).

§ 7.3.2 The problem of designing neighbourhoods

Defining a neighbourhood and what a neighbourhood should contain is a common and controversial research topic. In the development of Model B we implemented the features that are generally considered common ground for planning neighbourhoods, based on consistencies found in the specialist literature. These consistencies were used to structure the implementation of Model B:

- Many authors agree on a range of 5,000-10,000 people per neighbourhood / community as parts of districts with 20,000 to 100,000 people.

- This size contains enough critical mass to provide at least a school and a community building, a main street with shops, a central square and a local square.
- The neighbourhood should have mixed uses, both vertically and horizontally. The mix increases as it comes closer to the neighbourhood centre.
- Block size varies between 1 acre and 1 hectare. However, there are many successful alternative examples with bigger blocks (e.g., Berlin, Amsterdam).
- Squares present even wider variations. Smaller squares seem to be more frequently associated with the unplanned (informal organic) city and bigger squares with the formally planned (authoritarian / imposed) development. Although the amount of alternative examples is considerably larger than in the case of blocks, the same range of between 1 acre and 1 hectare is still acceptable.

(Alexander et al., 1977); (Barton, Grant, and Guise, 2003); (Marshall, 2005); (Moughtin, 2003); (Moughtin and Shirley, 2005); (Jacobs, 1961); (Steiner and Butler, 2007).

Marshall (2005) presents a large survey of urban grid types identified by several authors in their writings on urban morphology (see Appendix 4 to his book). The rectangular, radial and irregular (organic) grid patterns at least seem to be present almost everywhere in urban design literature. The two first were used as the main design drivers in Model B. The organic grid pattern should be regarded as an emerging pattern and therefore we did not use it as a design pattern. Marshall cites some characteristics of the common structure of street networks that are usually perceived of as pleasant urban environments. Such street structures have short and long routes, a relatively large number of 'T' junctions, some 'X' junctions and some cul-de-sacs. Marshall calls this the characteristic structure. In order to integrate these ideas, we added a third street pattern to our set of street grid types which we called the recursive street generator. This street pattern is described later and corresponds exactly to the UIP 011 – (grid by) *RectangleDissection* – see Appendix 2.

How do urban designers make decisions? – The design process workflow

As already addressed in this thesis, the design process proceeds through a series of reflective moves implying negotiation between the problem and the trial solutions using analysis, synthesis and evaluation. In developing Model B we tried to improve these characteristics in comparison with Model A.

The structure of design problems contains determined components that can easily be computed and also undetermined and underdetermined ones which are intrinsic to design problems (Dorst, 2004). In urban design, density indicators (Berghauser-Pont and Haupt, 2010) fall into the category of determined components and can easily be incorporated into a design tool, thereby providing continuous updates on the measures as long as a consistent geometrical model is available. The main goal in this implementation was to support the reflective features of design practice by providing continuous calculations by computational means to update information on determined components of urban design problems after each design move. Such a system should offer the designer precise information on indicators and the properties

of the designs being produced, thereby allowing for the interactive manipulation of input as a reaction to output.

On the basis of these considerations, Model B should:

- be applicable, regardless of the design context and district size, to a fairly large variety of design programmes;
- be interactive and responsive, providing good visualisation output both in terms of design layout and the associated analytical data (indicators, attributes, indices, etc);
- be able to implement and design the main features that compose a neighbourhood;
- provide data for the elaboration of the plan's regulations.

Taking these aspects into consideration we implemented a parametric urban design system using a NURBS-CAD environment and a parametric programming interface. The CAD environment was Rhinoceros® and the programming interface was Grasshopper®. The system aims to design urban plans at neighbourhood or district level, to use Berghauser-Pont and Haupt's terminology (2010, page 103).
A particular kind of urban design workflow was considered in this model, starting from a specific set of initial inputs (geometric and data inputs). Figure 41 and Figure 42 show the geometric and data inputs. Figure 43 shows the main outputs of the model. The design process follows a particular workflow organised as a specific set of procedures:

1. preparation of the design base (the analytical phase developed on any preparatory platform, preferably a geographic information system) – this includes the identification of areas which present difficult topographies for design implementation;
2. insertion of geometrical inputs, associating them with specific design patterns;
3. insertion of data inputs, some of which can be considered goal inputs;
4. design exploration by manipulating options, primitive geometry transformations and parameter inputs;
5. confronting the modelled design with the generated data;
6. distributing a programme of uses in an accepted layout;
7. using the plan and the generated data to define regulations for the plan at island level;
8. mapping the plan onto the topography.

There are two types of inputs – geometrical inputs and data inputs. Figure 41 shows the set of geometrical inputs, which is divided into 4 basic types: the site (defined by polygons); the composition elements, which are subdivided into main streets (defined by lines and curves) and focal points representing the location of the neighbourhood centre, local squares, public buildings and city objects in general; a vertical parameter to define the maximum number of floors; a set of grid types (rectangular, radial and recursive). Each of these inputs has a set of associated parameter inputs. A main street, for instance, has the attribute *street width*.

The elements are located anywhere inside the site boundary by the designer and may be relocated at any time during the design process simply by dragging them elsewhere or changing them by moving their grip points. A selected grid type (rectangular, radial or recursive) is tailored to the site boundary and the algorithm combines the grids with other urban elements, filtering them out from the main grid geometry. The filtered parts may correspond to different components of the design (e.g. public buildings) and for this purpose the filtered geometry is manipulated with specific inputs which are associated with the filtered parts. Some inputs refer to options, which are defined as switches in the design interface that allow the designer to choose, for instance, which grid type to apply. The other input parameters allow the output appearance of each type to be changed.

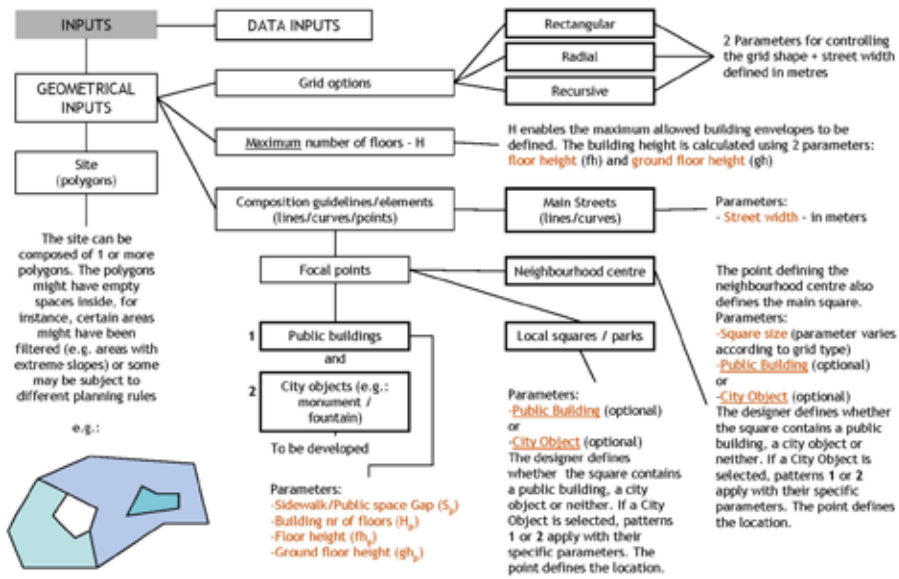


Figure 41
Geometrical inputs.

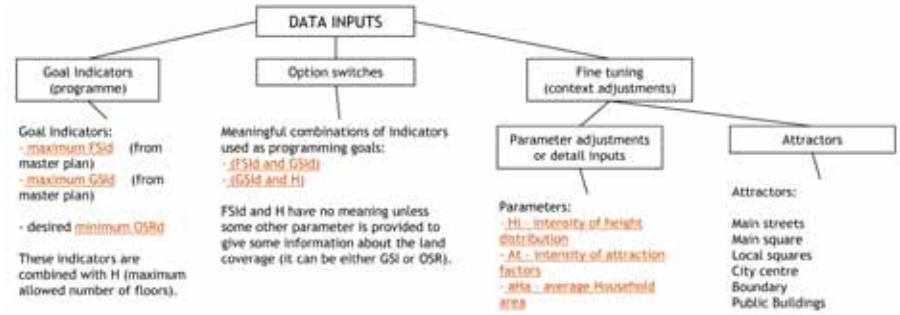


Figure 42
Data inputs.

The data inputs are shown in Figure 42. There are 3 types of data inputs: goal indicators, which define the density goals or constraints of the urban programme, option switches, and context adjustments which are input parameters regarded as customisable parameters that can be manipulated by the designer to fine-tune the design. The designer explores the options and parameters following the workflow sketched in Figure 43. The designer considers the topography and the contextual conditions of the site at the beginning of the design process, extracts the areas that are considered too steep and eventually identifies zones to which different planning rules or strategies should be applied. This can be a traditional analytical process or it can be enhanced using GIS tools to provide the information required to set the primary elements of the design. As a final result a site area subdivided in several zones is obtained, to which different planning strategies may be applied (see also Figure 41). The site geometry is inserted into the design environment. The designer defines the position of the main composition elements of the design, namely a focal point, and the main streets within the site boundaries. This is drawn directly in the NURBS-CAD drawing environment. Accuracy in terms of the composition of elements is not an issue, as they may be moved or changed at any time during the design process. The designer then chooses between three available grid patterns (rectangular, radial and recursive) and explores variations with the available parameters. The selected grid is dynamically updated in the drawing interface. The orientation of the grid is a common parameter which allows the designer to change and fine-tune the grid orientation.
The recursive street generator designs street grids based on a rectangle dissection rule (Figure 44). The rule applies if the area of the rectangle (i.e. the block) is bigger than a user-predefined minimum area and the block's sides are constrained to be bigger than a minimum value that corresponds to a user-defined number of 'pixels' or basic modular unit. The 'pixel' size is also predefined by the designer to define the *grain* of the grid. 'Pixel' size plays an important role since it enables the chances of having more or fewer 'T' junctions to be controlled. Basically, the bigger the 'pixel', the greater the chances are of coinciding subdivisions in two neighbouring rectangles, therefore

reducing the number of 'T' junctions. Finally, the rectangle proportion is also controlled. The available parameters allow the grid character to be shaped. Examples of the three grids can be seen in Figure 43.

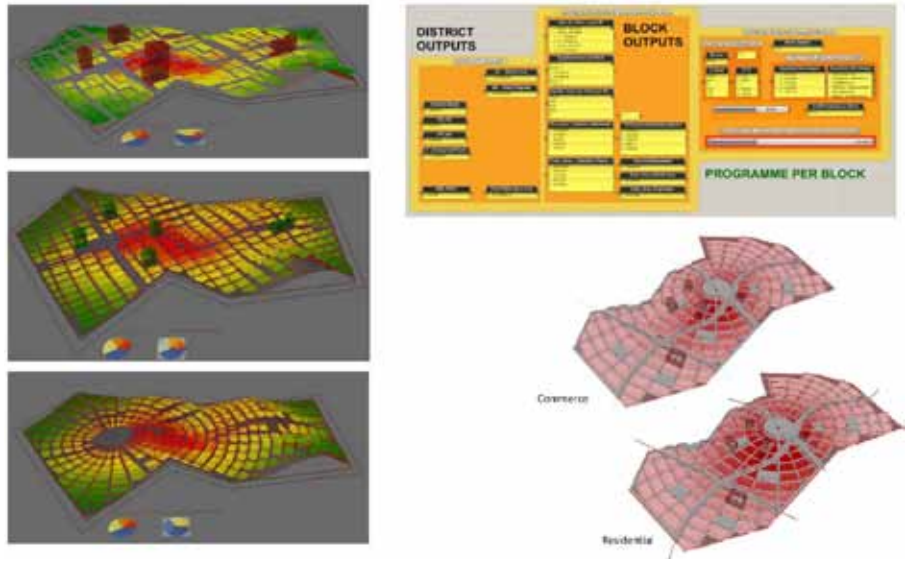


Figure 43
Outputs of Model B. The three images on the left show the three available grids. The upper right corner shows the data output interface in which density indicators are shown at district scale, block scale and per block. The lower right corner shows the distribution of commercial and residential use in the plan.

The designer filters several areas out in the main model, to which exceptional rules or design goals can be assigned. For instance, certain blocks can be set to be small public squares and others to be filled with public buildings. The main process is defined as a filter, a design pattern that isolates particular sets of geometric features from the main geometry. Different parameters may be assigned to various sets of public buildings filtered in this way. The process can be replicated to create sets of geometry parts to which different generation rules can apply.
The building height is managed by setting the maximum allowed number of floors. This value is used as input for the 3D representation of the maximum allowed building envelope. The number of floors is defined as a target number which is distributed throughout the grid as a simulation of land value. To simulate the effect of land value we defined the number of floors in a block as a function of the distance to a set of positive attractors – main square / main streets / the city centre – and a set of negative attractors (repulsion effect) – the site boundary. This function changes the number of floors depending on the resulting calculations. The distribution is determined using bounded distribution methods in which the bounding condition is a target density defined in the urban programme. The designer controls the intensity of the overall

attraction effect using a set of sliders. The indicators are calculated and provided in the interface, allowing for immediate empirical evaluation of the output. The evaluation can be further supported by hints provided by studies on morph-types such as the studies developed by Berghauser-Pont and Haupt (2010) on the relation between density and urban morph-types. The indicators also follow the conventions of Berghauser-Pont and Haupt (2010), allowing for rigorous application of their measuring theory. The basic indicators are: FSI_d – Building Intensity; GSI_d – Coverage; ND – Network Density; and OSR_d – Spaciousness. The indicators are calculated at district level.

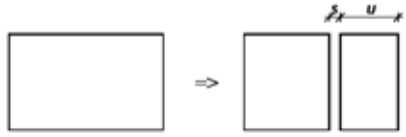


Figure 44
Rectangle dissection rule

Following density distribution an uneven distribution of building intensity per block is obtained. The system also outputs block level calculations of density indicators, thereby providing the designer with extremely accurate information on density indicators, both at block and district level (see Figure 45 and Figure 46).

Outputs at block level	
Identifier	Unit
Block number - index	Integer
Attributes / urban indicators	
FSIi - building intensity	m^2/m^2
GSIi - coverage (block)	m^2/m^2
GFAi - gross floor area (per block)	Square meters or hectares
OSRi - spaciousness (private OSR)	m^2/m^2
Hhi - number of households in block	Integer
Mi - block construction mass	m^3
Li - average height (block)	Number of floors
max H - maximum height	Number of floors
Ai - block area (extracted from geometry)	Square meters or hectares
Distributed function / use	
Function	String
Function intensity in block (%)	Percentage
Area per function	Square meters
Geographical location	
Centroid of block polygon	Point coordinates
Polygon corner points	List of point coordinates

Figure 45
Outputs at island (block) level. The shaded indicators are interpreted as defining the base for a regulation or implementation code at block level.

Outputs at district level	
l - Network length	M
Nd - Network density	m/ m^2
Connectivity Ratio	Crossings per hectare
GFAd - gross floor area (district)	Square meters or hectares
GSId - coverage (district)	m^2/m^2
Hhd - total number of households	Integer
Md - total construction mass	m^3
Dd - density as households per hectare	Hh / hectare
B - base land area	Square meters or hectares
Ld - average height (district)	Number of floors
PPI - parking performance index	If ppi>1 parking capacity is insufficient
DF_e - daylight factor	0-100
DPI - daylight performance index	% of floor space exposed to sunlight

Figure 46
Outputs at district level

Typically, master plans use a density measure expressed in terms of the maximum allowed density (defined here as FSI_d), a maximum coverage, and quite often a maximum allowed height or number of floors (*NF*) as planning devices. In most situations, in order to foster design flexibility the values express maximum allowed limits which are set independently of each other. A minimum amount of free public space (minimum allowed *OSR*) can also be used as a planning device to express certain desired qualities for public space. Considering the given site area, the existence of a maximum allowed FSI_d provides for the calculation of a maximum allowed gross floor area (GFA_d) for the district. The average number of floors, *L*, is calculated as an output at district and island (block) levels. The system outputs regulations defined at island level in terms of the maximum number of floors (*NF*), maximum *FSI*, maximum *GSI* or minimum *OSR*. Figure 45 shows all the system outputs at island level and Figure 46 the outputs at district level. The 3D model shows the graphic outputs of the indicators for the designer's visual assessment. Any change applied to the model, including filtering blocks to define squares, parks or public buildings, will automatically update the outputs for each block.

Land use is addressed in the model as a simulation. It simulates the distribution of uses considering the same urban attractors cited before but using an independent interface. The idea is that attraction in terms of land use is influenced by different phenomena to density attraction and in any case responds differently, depending on the kind of use. For instance, the distribution of commerce behaves differently from small industry. The attraction/repulsion effect is set differently for each use, considering the land use programme in terms of the relative percentages of Housing, Commerce, Workspaces, Facilities and Small Industry. The model outputs a visually defined distribution of uses per block with a pie chart and a block indicator.

The use of simulations in a real case scenario should be subject to critical interpretation. Planning mixed use is a complicated issue. It is consensual that the best urban areas have mixed uses. However, it is almost impossible to know the right proportions for each use, as this is context dependant and naturally dynamic over time. The goal of the system is to produce a regulation at district level and a regulation per block. In all cases, the regulatory system should include a means of managing the system dynamics. This can be achieved for instance by using the programme formulation interactively to set somewhat flexible goals indicating a range of solutions per use instead of rigid goals and single solutions. In scenarios like this, the implementation should be regulated in order to maintain its particular interactive dynamics but constraining it to guide solutions towards the desired goals. Allowing fluctuations within the defined range is one possible strategy. Allowing a trade-off between blocks is another possibility for maintaining the natural dynamics. Nevertheless, the simulations allow designers and stakeholders to be aware of possible scenarios and their characteristics in terms of the possible mix of uses.

The design solution can be remapped onto the topography. The main principle is that there will not be any transformations in the grids because the extreme sloped areas were filtered out at the beginning of the design process. San Francisco was the main

driver behind this decision. However, the main streets may still be deformed and readapted to the topography following geodesic curves.
The system provides a very empirical design interface which allows designers to see the consequences of the design moves in real time. As an added feature, the designer is able see the urban indicators shown in Figure 45 and Figure 46. The relationship between the design morphology and the urban indicators is therefore immediately available for designer control and reaction. The flowchart in Figure 47 summarises the system's structure[33].

§ 7.3.3 Discussion and future work

Model B is structured as a very interactive design system providing the most common features for designing neighbourhoods. The system allows for the parametric manipulation of a design whilst providing density measures for the outputs for a better assessment of the qualities of the proposed design. This assessment was based on commonly used urban indicators. The output is both visual and numerical, extending the designer's awareness of the consequences of his/her design decisions.
However, the tool still contains several limitations, most of which may be solved with further work or by developing the technology. The main tasks for future work are:

- Adjusting the standard grid to an existing context, connecting new streets to existing ones, and connecting streets from the grids of two neighbouring zones.
- Extending the possibilities of adjusting the design to the topography.
- Improving environmental issues, even though the designer is able to assess daylight and parking performance indices and control the grid orientation.
- Extending the tool to include property features, as an important upgrade. The parametric model has good potential for this purpose because it allows all related operations to be connected. Considering the output of Model B as a plan with regulations at block level, including property subdivision information would complete the information on regular urban design.

33 The flowchart corresponds to the model defined up to April, 2011. The model has since evolved into a different structure.

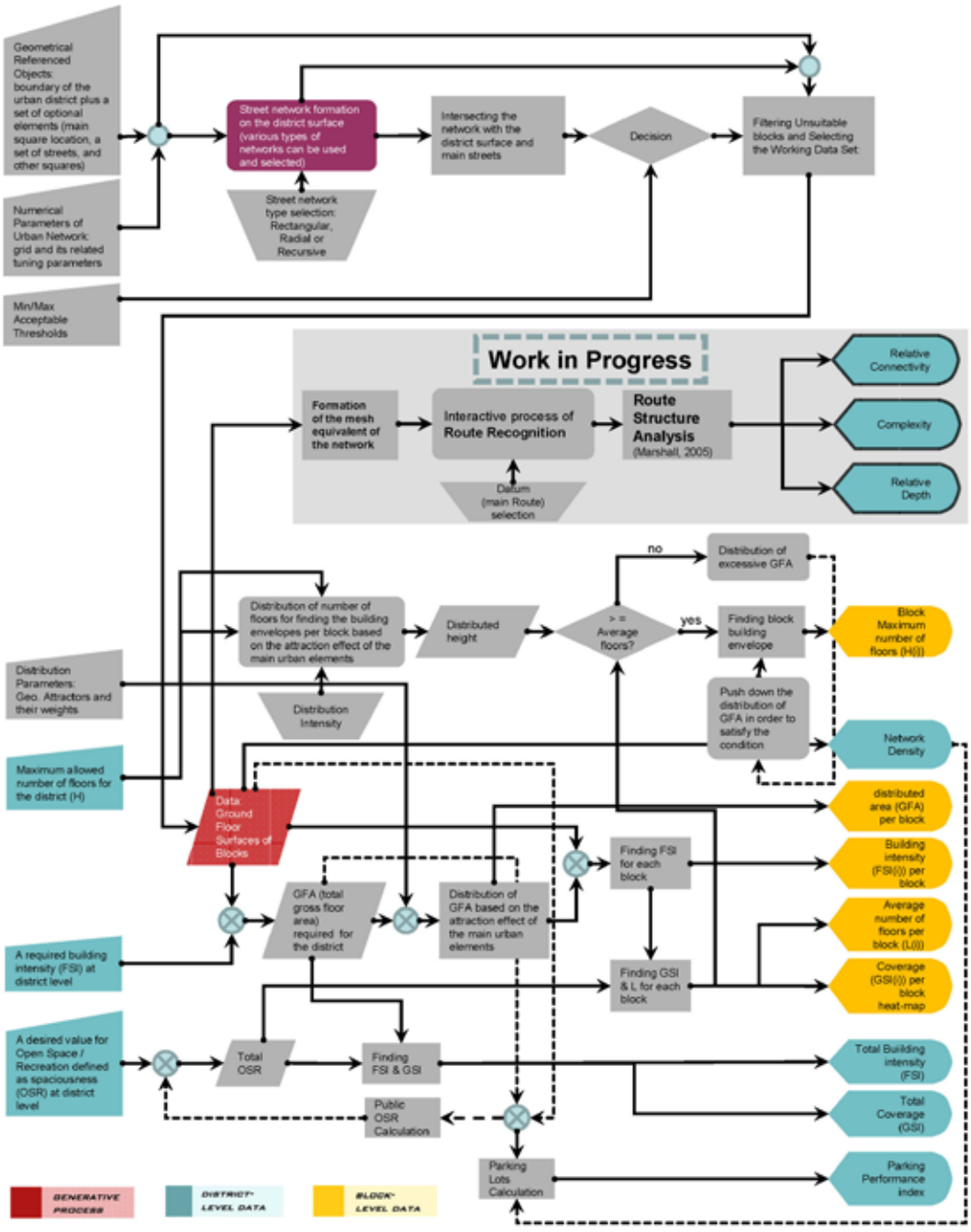


Figure 47
Flowchart of the urban design tool

- Integration with GIS tools or at least with GIS supported analytical routines. The advantage of the programming environment used is that there is no data loss. The parametric model contains all the data produced at any moment in the generation process and this data simply needs to be extracted. However, retrieving data from the GIS is also an important feature that should be improved in terms of directly inserting the results of analyses.
- Finally, the system may be very slow if the geometrical operations become very complex. The method used to address this problem until now has been to limit the model structure to the lowest possible amount of geometric calculations. However, the amount of geometry involved is very much dependent on the particulars of the design context and it may not be entirely possible to reduce it to very short calculations. It is likely that complex designs may take a few minutes to update each design move.

However difficult it might seem to solve some of these limitations, we should stress the positive achievements of the design system:

- The system is able to accommodate typological solutions for neighbourhood design which respond to a large number of urban design problems of this nature.
- The system is very dynamic and interactive, allowing for continuous design exploration by updating solutions which other inputs change.
- Every design update is expressed both in terms of visual and numerical output (density measures and derived indicators) enabling designers (or any decision-maker) to be aware of the implications and qualities of a particular solution option.

The current approach focuses on the internal relationships between elements that compose neighbourhoods. The geographic information for the urban context is currently missing. An overall approach to urban design needs to take into account the spatial associations between the density and land use patterns of a neighbourhood and its context. In the next developments we plan to integrate geo-referenced analytical methods to include contextual information for integrated urban design decision-making. In addition, the effects of the proposed design on other urban areas also need to be evaluated as part of the whole urban network.
These goals are easily achievable in a design process that follows a sequence of analysis using a GIS, then imports the necessary data and geometry from the GIS into the design platform and undertakes parametric design exploration by comparing the design with density measures, and finally feeding the information back to the GIS to evaluate the results using GIS or GIS-based analytical tools. This sequence should be applied in cyclical procedures every time the evaluation is not satisfactory and feedback is needed in the form of new analytical information captured during the process. The best solution would be to develop extensions of this model to perform the equivalent

integrated analysis directly on the evolving model in order to maintain the best possible quality of design model.
As an example of this kind of approach, a route structure analyser inspired by Marshall's route structure analysis (2005) is being developed for incorporation into the regular design workflow. The analyser is already operative in the generation of the recursive grid, as the generation process directly builds a route structure. However, when the grid is inserted into the main geometry the route structure is subverted and immediately raises interpretation issues. In order to solve the interpretation problems identified by Marshall, an algorithm could be developed to provide optional interpretations which the designer can choose manually by selecting the desired interpretation. However, developing such algorithm involves specific research.
In order to improve the usability and adaptability of the tool for specific practical situations most of the functions could be organised as design patterns, as proposed by Woodbury (2010). Figure 41 shows some concepts indicated in boxes framed with a thick borderline. The concepts in these boxes can all be defined as design patterns: an identical template code which can be reused for performing recurrent design operations. In Model B these design patterns correspond approximately to the urban induction patterns presented in the previous chapters.
The use of design patterns is envisaged as follows. Consider a site involving two different polygons, for which different master plan regulations were defined upstream; one area, for instance, is high density and the other low density. A small area in one of the polygons is isolated in advance and delimited as having a steep topography. The designer identifies several important streets in neighbouring areas of the city that should connect to new streets in the intervention site, using the design pattern – Main Streets. If the streets are envisaged as having the same width (as well as other regulations) the same pattern can be used for all the streets. However, if the designer wants them to be different (for instance, because the streets they are connecting to are different), then s/he will need different Main Street patterns. The designer draws a curve in the drawing environment corresponding to the street centre line and assigns it to the pattern. A slider belonging to this pattern controls the street width. In the case of equal street widths, multiple geometries (curves) can be assigned to the pattern. A similar procedure can be used to define Local Squares, Public Buildings, or City Objects, although the input geometry in these cases is a point. However, these patterns can only be applied after the grid is generated because these elements of the design are applied to filtered parts of the grid. Grids also work as a design pattern. This pattern contains three options – the rectangular, radial and recursive grid – which can be selected with a switch. The pattern can be assigned to a polygon or to multiple polygons. However, when assigned to multiple polygons these areas will all have the same grid orientation and the same parameters. Conversely, a designer can assign a different grid to each area allowing for the use of different grid types or simply different grid parameters. The design can become quite complex in this way. The main operations for the grids (Main Streets, Local Squares, Public Buildings, or City Objects) all use a hidden design pattern which can be explained as a filter. The common part of these design patterns is a

smaller design pattern, a filter which extracts parts of the grid geometry and assigns different parametric rules to it. The filtering process is also the process which keeps track of changes and therefore continuously updates the density measures.
The final version of the three modules in the City Induction project, namely the formulation, generation and evaluation modules will include some of the goals defined for future work.
Finally, it should be noted that Model B does not implement block types. It was developed as a tool that designs large scale (district / neighbourhood level) urban plans designing the grids and the main features of a plan of this scale, and outputting solutions in which *islands* are the minimum elements. At this level the output takes the form of a regulation to be applied in more detailed developments. Such an output could then be provided for different designers to design the blocks or the buildings for the blocks, each following their own design convictions. For more accurate work with block regulations, a comparative study such as the morphological studies developed by Berghauser-Pont and Haupt relating density indicators and urban morph-types should be able to support decisions by establishing more accurate relationships between morph-types and density indicators. Furthermore, such a study could be further enhanced by establishing relationships between (A) building types, (B) their uses and (C) density indicators. If a broad sample is gathered using statistical methods such as data mining for pattern finding, it may be possible to find relationships between specific combinations of A, B and C variables and relate morph-types with qualitative behaviour. Such information, if inserted into the model, may give the designer real-time values for his/her moves.

§ 7.4 Comparing models

Two different prototype models for urban design generation were presented in the previous sections. They are incomplete, but both show singular qualities intrinsic to the implementation approach followed in each case. In terms of conceptual approach, the two models can be summarised as follows.

Model A – the implementation follows the theoretical model defined in Chapters 5 and 6 quite accurately. The main concepts envisage the integration of programme formulation, generation and evaluation in the context of the City Induction project. As such, although working autonomously, it has been developed in such a way that the initial components of the design can be captured from the ontology being developed by Montenegro et al. (2011), including programmatic specifications. The design results would be generated in GIS compatible formats, i.e. thematically layered according to the ontology of the design process (see Figure 9 and section § 5.4 , page 92), with each

layer containing only geometry of the same kind, such as points, lines or polygons. The data generated is also stored in a database and can be reused in the GIS environment.

Model B – takes advantage of the characteristics of a parametric NURBS-CAD environment and focuses on the tool's usability and interactivity with the designer, in terms of both the geometric model and the data flow. The data flow considered in the model essentially involves density-related indicators which play a role in the decision-making process, from both the designer's and the stakeholder's viewpoint (in fact from the viewpoint of any interested actor). Since 'spacematrix' theory (Berghauser-Pont and Haupt, 2010) provides many derived indicators that can easily be calculated from the geometric model, several qualitative indicators may be provided, allowing designers to improve their awareness of the results and consequences of their design moves. The main idea of Model B is to explore relationships between design and density indicators based on the concept that most of the information the designer needs can be captured directly from the geometrical model being designed.

The following table establishes the main comparisons between the two models.

Model A (CAD-GIS model)	Model B (parametric model)
Integrates CAD/GIS functionalities into a single design environment. The AutoCAD environment facilitates the import and export of designs and associated data. The model provides an accurately structured prototype for this purpose but the implementation is quite extensive, both in terms of complexity and the time consumed in programming tasks.	Lacks integration with GIS platforms (can be linked, but using an external platform).The Grasshopper model keeps all the data that can be extracted or derived from the geometric model. It also allows this information to be stored in a database or Excel sheets, as well as retrieving information from similar bases. This was actually the first solution applied in the development of the recursive pattern. As such, GIS integration may be accomplished in a similar way to Model A.
It is extendible (extending the software using the API) – but this is hard. Extendibility in AutoCAD is extremely versatile due to its various APIs but depends essentially on hard coding regardless of the language/API in use. Programming and extending the model is very time-consuming but the development is structured consistently if the theoretical model is respected.	It is extendible (extending the program in Grasshopper) – and this is easy. This interface is extremely user-friendly in terms of extendibility and customisation. The results of what is being programmed are immediately shown on the drawing interface. This makes the system very easy to debug and test. The only problems regarding this environment are: (1) the lack of recursive functions in Grasshopper and (2) the amount of geometrical operations, which can make the system too slow.
Stores data in a database / adds data to the GIS database.	Provides all the data generated (can be linked to a database)
Feedback loops imply re-generation. It was planned in the theoretical model to register all actions in the database – as a history of actions (Beirão et al., 2009). However, this feature has not been implemented yet.	All inputs can be changed at any moment (the model updates in real time). This is the main advantage of Model B over Model A.
Easy integration with tools running in the GIS environment (any kind of assessment tool from simple queries to spatial analysis – e.g. space syntax, place syntax). However, GIS analytical processes and CAD changes are independent processes. Analysis can be performed only on finished designs.	Integration with analytical tools and GIS tools still needs to be developed. It is expected that there will be more difficulties when integrating the CAD model in the GIS environment. However, its practicality should be similar to Model A, unless specific analytical processes using the parametric model are developed for the same purpose. The model already performs some evaluation tasks, as some qualitative indicators are automatically calculated and constantly updated.

In theory the model can always be extended to widen its flexibility range. This process depends exclusively on the creation of new UIPs. The limitation regarding extendibility is the fact that the UIPs are hard coded and take a long time to programme. The development of a customisable meta pattern structure would contribute greatly towards better performance in terms of tool extendibility.	The model can always be extended to widen its flexibility range and field of application. This process depends on the creation of new design patterns. The main limitations of model B regarding extendibility are related to its requirements in terms of computational power. If the geometric model becomes too complicated it is also likely to become too slow.
The design space variability is dependent on the amount of possible UIP conditions, as well as their options and parameter variability. Any specific urban grammar is capable of generating many solutions within the design space of the grammar.	The design space variability is dependent on the amount of different available design patterns, as well as their options and parametric variability. Once the design patterns used in a design are set, the variability is limited to options and parametric variability. Although options and parameters can be changed at any time, the system produces only one solution. However, changes on the level of options and parameters can be effected almost in real time depending on the amount of geometrical operations. Fundamental changes based on changes to primary elements of the composition – for instance changing the number of main streets or number of zones - are difficult to accomplish.

§ 7.5 Discussion of results – validating the models

The main practical differences between the two models become apparent when they are confronted with an objective validation procedure. A correctly structured validation should encompass the following goals:

1 A demonstration that the theoretical model is adequate by showing that:
   a The grammars in the theoretical model are able to generate the case studies.
   b The grammars in the theoretical model generate many different plans for different contexts.

2 The theoretical model is extendible and customisable.
   Evidence that the prototype implementations are able to:
   a Generate the case studies.
   b Generate other plans in other contexts.

3 An appraisal of the prototype models by practitioners.

4 Application of the prototype models in real case scenarios.

These goals are discussed below.

1 – The first point is partially demonstrated in the thesis in Chapters 5 and 6 when describing the theoretical model. The argument regarding the broad application of the theoretical model is further supported by the examples shown in Appendix 3. The argument regarding extendibility is clear in the block type generation patterns, both in the theoretical model and in Model A (see Section § 6.3 , page 152, and Section § 7.2 , page 192). Model B's extendibility lies in the reusable design pattern structure of the system, which can be copied to adapt or extend the model as required by the complexity of the design problem. The design pattern structure can be improved and extended. Model B is currently being used and extended by a Masters student in a real case application, namely in the study of future alternative scenarios for an industrial area in the Sintra region in Portugal.

2 – Demonstrations involving the prototypes are confronted with the prototype limitations. For instance, Model A does not generate squares yet. However, the features implemented so far generate the equivalent parts of the case studies and other plans, as shown in this chapter and in the examples in Appendix 3. The accurate structure of Model A requires a large team of professional programmers to achieve all the goals defined in the theoretical model. Most of the time consumed in programming Model A is spent writing instructions for storing and retrieving data from databases or simply managing the data in these bases, that is, clearing and updating attributes or variables changed by the rules or in programming adequate interfaces. In any case, in addition to this chapter, Appendix 3 shows some plans already generated by the two implementations. It should be noted that this does not simply involve the generation of geometrical models but also the generation of the semantic attributes of the geometry and density indicators for the generated design. Nevertheless, what Model A lacks in terms of density calculations and appropriate interfaces, Model B lacks in terms of the generation of semantic attributes. However, it should be stressed that the latter is compensated by the extensive designer interactivity of the model.
Appendix 3 shows several applications of Model B, presenting the generation of design solutions for several different contexts. Most difficulties in the use of the model in real situations concern the adaptability of the main representations to the existing situations. For instance, linking an existing organic street structure to a new layout and maintaining street continuity can involve several problems. Likewise, the results of topographic analysis may need a specific method to incorporate them into the plan. However, although sometimes a little time consuming, most situations can be solved without great difficulty in Model B. Implementations using Model A frequently showed the need to develop new urban induction patterns which have to be encoded from scratch in VBA or VLisp. As such, Model A still reveals some difficulties in terms of practical use. Nevertheless, it should be stressed that its accurate structure, if further developed, seems to be a more adequate approach to the implementation of urban design software as an extension of AutoCAD Civil3D features. Model A therefore shows a simplified proof of concept of what an urban generation tool embedded in AutoCAD should be.

3 and 4 – The first confrontation of Model B with a real design problem was carried out by its co-author, Pirouz Nourian, a few months after the implementation had begun. Nourian tried to use the model that had been developed so far but found the software structure too rigid to adapt to the design problem he was facing. The entire software needed adjustments to solve the new design problem. However, the main concepts were not questioned, and specifically the real-time calculation of density indicators always seemed an important approach that called for the attention of a few researchers, including Meta Berghauser-Pont, one of the authors of Spacematrix (2010) which provided the theory for the calculation of such indicators. At the time, Nourian said that Model B should be restructured into a pattern-like structure as in Model A in order to improve the modularity of the system and therefore its usability. Such improvements were later introduced, following Woodbury's design pattern concept[34].

The models have been shown to a few practitioners including Chuva Gomes, the author of Plans 1 and 2, and Frits Palmboom, the author of Plan 4. Pak and Verbeke (2011) cite questionnaires and interviews as some of the most common and cost effective approaches to tool assessment. Some interviews were carried out, including one questionnaire in which designers could offer their opinions on tool usability. The main idea was to identify the most important characteristics of the proposed models, their main qualities and their inadequacies by gathering criticisms and suggestions for future improvements.

Both Chuva and Palmboom showed a greater interest in Model B than Model A. This was due to two factors: firstly, Model B is more developed and offers a larger amount of functionalities, and secondly, the interface and interactivity are a lot more user-friendly, offering a better view of the concepts involved in the models. The best reaction of both designers was their astonishment at the continuous updating of density indicator calculations. Palmboom made an important comment to the effect that the calculations were not always needed but both agreed that it was useful to have them available for consultation. Many other designers have praised the calculation features and the possibility of continuous updating during design exploration[35]. Whether they are used or not in the decision process is a matter for the designer to decide.

34 This part of the work is still in progress. Every pattern needs, on the one hand, to be simplified to the simplest possible structure in order to reduce computational resources and, on the other hand, to be as generic as possible, implying that it should support all the typical design decisions covered by the pattern as far as possible. Therefore, both implementations showed that this process implies a continuous improvement of patterns, by simplifying them whilst extending the range of their application.

35 In addition to Chuva Gomes and Palmboom, Model B was also shown to Meta Berghauser-Pont, António Castel Branco, Cristina Cavaco, Patrick Schirmer, Burak Pak, Tiago Trigueiros, Paulo Pedro and Sónia Taborda. All of them are professional urban designers involved either in planning practice, research, or both. Each time Model B was shown to designers, their first reaction was usually sceptical and most questions tended to compare the model with traditional ways of drawing urban plans. However, in every presentation to designers, the moment they first saw the calculation update after a change in the design, their expression changed instantly, usually followed by a smile or an astonished look. Although some designers still identified some minor issues after reflection, the initial positive reaction was detected in every designer.

One of the most curious comments came from Frits Palmboom. In reference to the grid generation patterns, he said that he was not interested so much in the possibility of generating grids but in placing axes (or streets) one by one in meaningful places, even considering that they may in the end become a regular grid. From his point of view, the grid was not a goal in itself but a consequence of a set of individual design decisions that ended up forming a grid. After hearing this comment the importance of one already available feature became obvious: the possibility of generating a grid by accumulating individual UIPs that generate axes. Improvements made to the models after this interview included improving the possibilities of designing grids in a less automated way. The advantage of such a possibility is the gain in reflective action, which can be exercised move after move. This also allows the direction of some axes to be manipulated locally, for instance, by generating a quadrilateral grid with one or more axes slightly rotated or connecting two important landmarks. It also solves two main problems related to the automated grids: (1) linking two grids generated in two adjacent areas, and (2) eliminating the algorithms for deciding which parameters to apply at each iteration during grid generation. In each case, density measures are produced, enhancing the quality of the information available and therefore improving the designer's awareness of his/her decisions.
Chuva Gomes mentioned that the tools showed precise data for measurements that are usually perceived empirically by the designer. He said that most of the time an experienced designer might have quite an accurate perception of the measurements involved in the design problem. This argument was also supported by Palmboom who added that at the beginning of the design process his experience would be enough to provide him with information on generic measurements without deviating too much from reality. However, the tool seemed to be very helpful in fine-tuning the solution because it allowed for accurate measurements whilst maintaining the possibility of further manipulating the design solution. In addition, it can be argued that the calculation tools may be very helpful for less experienced designers, even in the early stages of the design process.

Two important conclusions could be drawn from the interviews:

- The clearer the involved processes are, the better. Designers want to understand what the tool is doing and react strongly to 'black boxes', especially when an automated decision seems to overlap with what is considered to be a typical designer decision.
- However, having accurate information on the plan's characteristics, namely density data, is seen as a positive feature of the design environment as it improves awareness of the properties of the proposed solution. The designers interviewed agreed that decision-making could be improved as a result of such awareness.

One of the questions included in the questionnaire asked for suggestions for improving the implementations. Palmboom commented that after fine-tuning a layout, a designer works on the street profiles and therefore suggested that it would be interesting to find similar features in the software. An identical suggestion came from the architect Paulo Pedro in Portugal. This suggestion is very interesting and easy to implement following the street description (**SD**) and street component (**SC**) classes in the ontology (see Table 4, page 101 and Table 5, page 105). It is already part of the planned future work and is reasonably easy to implement.

§ 7.6 Recommendations for the development of City Information Models (CIM)

The question of whether the concepts developed in this thesis are extendible to architectural design emerged frequently during this research. BIM is, in fact, the structure in question and as such the answer is yes. The argument of the City Induction research team is that the structure proposed at the level of the City Induction goals is, in fact, a City Information Model (CIM), for which CItyMaker represents the design environment. Whereas in BIM we find an ontology modelling concepts such as walls, windows, rooms and their components, in CIM we find an ontology modelling axes, streets, their components, blocks, buildings, plots, and so on. In CIM, in order to avoid the criticism of BIM objectors regarding the difficulties in dealing with the first steps of design decision, CItyMaker introduces axes as the first compositional elements, using an existing feature of geographic information systems. In GIS, axes are the representations of thoroughfares at the lower level of detail (LoD1). As structured in this thesis, the CIM design environment contains additional generative features that provide the design system with tools to deal with flexibility that are particularly apt for the design of flexible urban systems.
The development of design environments for City Information Models aiming to create user-friendly urban design environments should encompass the following goals:

1. Understanding the urban design process and how designers work;
2. Compatibility with GIS systems;
3. Retrieving information from GIS and storing designs and design driven information in GIS;
4. Providing the maximum possible information, whether visual, qualitative or quantitative, in order to support design decisions.

Taking these goals into consideration, the following paragraphs define a set of recommendations for the development of design environments for City Information Models.

Recommendation 1 – Define a GIS compatible ontology modelling the concepts involved in the urban design process. The structure of this ontology is shown in Figure 9 and further developed with detailed insight into the street system described in Section § 5.4 , page 92. The ontology should contain the concepts involved in the urban design process and not simply a descriptive structure of the urban environment. Such concepts should also involve the base measurements for the calculation of indicators that might play a direct role in decision-making.

Recommendation 2 – The design environment should be based on generative features. Shape and description grammars are particularly suitable for this purpose because they provide generative behaviour that maintains a representational structure compatible with GIS environments. In order to provide this compatibility, the design environment should be based on compound forms of shape and description grammars to compute concepts (shapes and other components of urban concepts) taken from a relational structure of urban concepts, that is, taken from the ontology.

Recommendation 3 – Designers do not care about shape grammars or the related technical details. The generative features and design interface that concern them should hide those technical details and use a language common to urban designers. Therefore, design patterns should be used that encode urban design moves corresponding to common urban design instructions that can be understood by ordinary urban designers. The parameters and options available for each pattern should be also familiar to urban design practice and be meaningful in terms of design manipulation. The simplest way to present this idea is to call these design patterns typical urban design commands. In this thesis they have been called urban induction patterns to underline their generative properties.

Recommendation 4 – Define patterns as abstractly as possible, allowing for customisation as the means of providing them with designer specific language and avoiding the imposition of a design language. Patterns should preferably be defined with a very high level set of instructions common to an abstract design move. Personal interpretations of the design move should be captured by storing a set of user-defined instructions to fill in the gaps in the generic shape grammar code. The instructions stored capture a customised interpretation of the generic design move. Therefore, the design system needs to have a library of custom patterns which a designer might reuse whenever the context offers the opportunity to do so. Such a design environment is customised gradually, capturing the design language of the designer and providing it for reuse in further designs. It should be always borne in mind that designs consist of an arrangement of design moves.

Recommendation 5 – Provide information that can be captured from the model (the geometrical and data model). Density-based indicators are particularly useful for supporting design decisions. Berghauser-Pont and Haupt provide the best theoretical calculation model to support such calculations and additionally a set of derived indicators which are very informative in terms of supporting design decisions. All the indicators defined by these authors can be calculated from the geometrical model generated with the methods defined in this thesis.

Recommendation 6 – Provide a correct design environment, showing the geometrical model, visually processed information (e.g. colour coding for density), graphically processed information (e.g. the pie charts in Model B) and accurate calculations of all the useful indicators in an easy-to-read database. Provide fast access to the information, facilitating panning and zooming within the design interface and information queries.

Recommendation 7 – Make the model as reactive and responsive as possible, updating information for designers as soon as it can be captured from the geometry generated.

Recommendation 8 – The design environment should offer selectable design moves (UIPs) which progressively, move by move, form a design. Design moves should be programmed following the structure proposed in this thesis: moves are coded with compound forms of discursive grammars computing shapes and concepts found in the classes of an ontology describing the urban design process.

To summarise, the generation system has the following structure:

1 An ontology describes the concepts used in the urban design process. The ontology follows the structure indicated in Figure 9 and can be extended.
2 The generation system is defined by a very generic and broad urban grammar Γ defined by all possible arrangements of UIPs.
3 UIPs are compound forms of discursive grammars generating a design move. They contain grammars computing shapes and concepts found in some classes of the ontology. They are a sub-grammar γ of a specific urban grammar Γ' composed of a specific set of UIPs: $\Gamma' = \gamma_1 \times \gamma_2 \times \gamma_3 \times ... \times \gamma_n$ and $\gamma \subset \Gamma'$
4 A specific UIP uses only a few sub-grammars γ_i in which *i* refers the class in the ontology. A sub-grammar γ_i generates only one particular layer or type of representation related to the object class *i*, using a discursive grammar of the form: $\gamma_i = \{D,U,G,H,S_i,L_i,W,R,F,I_i\}$
5 The generation system generates representations of objects separated in layers. Each layer contains only one thematic representation (e.g. axes are generated by grammars computing shapes found in the ontology class **AN**) and one geometrical type only (points, lines or polygons).

The structure described above guarantees GIS compatibility and maintains the semantic structure of the system.
The recommendations described above should be followed in the development of the design environments for City Information Models. They provide a correct structure for GIS compatibility, generative design capabilities for the design environment and an information flow on the properties of the design being developed, thus enhancing assessment.

8 Discussion

The main driving force behind this research is the problem of planning and designing for the complex behaviour of cities. Flexibility is proposed as the means of dealing with complexity. The underlying concept is to promote flexibility in the design process and foster the design of flexible urban systems. This approach was suggested by several authors, as explained in Section § 4.2 - (Ascher, 2001); (Gausa, Hammond, and Hammond, 1998); (Friedman, 1997). The main idea is to abandon the traditional production of plan layouts, replacing this practice with the production of systems able to encompass changes in contextual conditions as well as changes in the implementation process. In other words, it should be able to support urban design as a negotiation process rather than a top-down decision process. Previous work involving shape grammar and pattern-based design suggested the use of such formalisms to support the goal of flexibility (Beirão and Duarte, 2009). This thesis presents a theoretical framework for urban design involving a design method and a computational tool based on the generative behaviour of shape and description grammars.

The central research question is: **what is the structure of an urban design tool that is capable of generating flexible urban designs for a given context and providing data that improves understanding of the design?**

As a response to this question the thesis develops the theoretical structure for an urban design tool based on compound forms of shape and description grammars that replicate urban design moves. The main concept is based on Schön's observations of design practitioners, identifying design synthesis as a series of design moves interspersed with moments of reflection on the results of each move. The design system proposed in the thesis is built on the observation that practitioners in the field of urban design share a large amount of design moves. It was observed in a set of case studies that these could all be generated from a small set of design moves. From these observations it was possible to identify a set of common design moves used recurrently by urban designers and infer their respective generative rules. They were called Urban Induction Patterns and they contain a discursive grammar which replicates the design move in different contexts. UIPs are generic design moves with a template code. This structure can be exploited by the designer by manipulating the available parameters and options displayed by the UIP. In addition, some UIPs are customisable. They contain a set of common rules and instructions for recording a set of customised rules which are stored in a library of pattern types for future application. UIPs are selected progressively throughout the design process until a design is completed. A whole urban

grammar is therefore synthesised from parts of the grammars of UIPs. The result is a customised urban grammar defining the flexibility space of the urban plan. The plan layout generated is an illustrative instantiation of the urban grammar that better expresses the designer's intentions.
This theoretical model was called CItyMaker. It was designed to support the implementation of an urban design generation tool. It also constitutes the generation model for the City Induction research project which, together with other two models, defines a tool for formulating urban programmes, generating designs for the formulated programme and evaluating alternative designs. In order to guarantee its independent use, CItyMaker was developed as a practical tool capable of designing urban plans independently of the other two modules. This means that the programmatic data can be acquired by the designer using any common tools available or any traditional means. Such information is inserted into CItyMaker as input to produce a design. Geometrical data is imported and alphanumerical data is inserted through the available interfaces. However, the production of a design needs continuous reflective action on the results of trial moves. Moreover, such reflective action is based on data associated with a layout rather than simply the layout. As such, the tool prototypes were equipped with an interactive interface that combines visual information on formal composition with other information useful for decision-making, namely information on density indicators and some performance density-based indicators. This structure allows the designer to carry out ongoing evaluations of his/her design decisions before considering a solution for more extensive evaluation procedures.

§ 8.1 Achievements and contributions

The main scientific contributions of the thesis are:

1 A theoretical model for an urban design tool involving generative design capabilities and an accompanying design method. The theoretical model provides a structure for urban design generation compatible with a GIS representational structure, including calculations on density-based indicators. The model provides a flexible design platform for the production of flexible urban designs. The flexibility space is defined by a specific urban grammar that is synthesised during the design process. Therefore, the thesis contributes to the field of computational methods applied to urban design theory.
2 An ontology describing the concepts involved in the urban design process, which contributes to the development of knowledge bases for urban design.

3 A shape grammar formalism for developing urban grammars, which contributes to the field of shape grammar studies.
4 A set of recommendations for developing software for urban design, namely in terms of how it should be structured to support GIS interoperability. This contributes to the field of computational methods applied to urban design.
5 A design method to enhance the quality of the information flow supporting design decisions in urban design processes. This contributes to the field of computational methods applied to urban design theory and to urban design practice. The method enables design decisions to be rooted in better grounded information.
6 A tool for supporting studies on the relationships between urban morphology and density. This contributes to urban morphology studies by improving awareness of the relationships between urban morphology and density.

The contributions made by this knowledge to design practice are likely to improve the quality of urban design, its management and response to complexity. In other words, the above contributions will allow for improvements to flexibility in the urban design process. Without introducing any other meaning to the term sustainability than the internationally accepted one [WS10], the approach proposed in this thesis will certainly provide a step forward towards the production of more sustainable cities, at least in the sense that it provides a greater capacity for designing cities that are able to adapt to the evolution of societies[36].

The improvement in data flow during the design process is also likely to improve the efficiency of participatory processes, in the sense that the proposed systems allow for alternative scenarios to be considered and supporting data provided for each scenario. Based on the improved information on the alternative scenarios, the stakeholders involved are in a better position to evaluate the consequences of the available alternatives. This is likely to improve the quality of decisions on how to shape our cities. From a social point of view the process can be both better informed and more democratic.

Regarding shape grammar studies, one of the main problems in research involving shape grammars is that it has failed to interest designers in using them in design practice. This is probably due to the fact that most research has focused on the use of shape grammars for analytical purposes and, in particular, the analysis of historical styles, e.g. (Stiny and Mitchell, 1978), (Buelinckx, 1993), (Flemming, 1987). Designers are usually interested in two things: (1) solving a design problem and (2) finding some innovative, expressive way of doing so. As such, a scientific domain that does not present strategies for producing creative design does not attract the attention of typical designers. In addition, with regard to problem solving, design problems may contain several determined components which are by definition computable components, but

36 Note that references cannot be made to city sustainability without considering both building and urban planning sustainability. They are interdependent.

many components of design problems fall in the category of undetermined and underdetermined components (Dorst, 2004). The latter components tend to be stressed by most designers as the main characteristics of design as a creative activity, unlike engineering problems. Undetermined and underdetermined components of design problems are difficult and, in most cases, even impossible to compute. The understanding of design problems is, as pointed out by Lawson, built up progressively from a process of negotiation with evolving solutions. A grammar therefore also seems too definitive a formalism to deal with this problem.

However, recurrent procedures can be identified in design practice and, in particular, in urban design practice. When interviewing Frits Palmboom, a brief summary of the research was given to him at the beginning of the interview including a short explanation of shape grammars. He was also told that the aim of the interview was to understand what his design rules were when designing Ypenburg. His first reaction was typical[37]. He immediately replied: '*I do not use any rules.*' The question was then rephrased by inquiring about his design methods. Frits Palmboom proceeded to explain that he always followed a similar procedure, starting with a visit to the site. After a while, to explain his first design move, he said: '*I always search for the longer line in the territory*'. He then, picked out some other designs to reinforce this statement and explained his interpretation of the same concept in other plans. Palmboom is absolutely aware of the recurring moves he makes. He sees them as a personal method but does not accept the use of the term 'design rules'. This can perhaps be explained by the fact that methods are perceived of as flexible and open to personal interpretation whereas rules are perceived of as strict procedures. Shape grammar rules are, in fact, strict in the sense that each rule is bounded by the limits of their particular parameters. The freedom of a grammar, however, is essentially defined by its versatility in combining rules rather than their parametric freedom. Nevertheless, a grammar defines a design language of some kind and most examples found in scientific literature include examples of classical styles. Designers find two problems in this: (1) they are not interested in designing using an old-fashioned design language and (2) they are interested in developing their own language, which means defining their own design rules. Designing using a given language is simply rejected by most designers, even if they end up following some conventions. It is therefore necessary to allow designers to feel this freedom.

The main perspective in this research is that if we provide the necessary space for designers to develop their design languages, assuming that they may be interested in exploring the generative potential of shape grammars, they may find new interests in their use. There are two interesting aspects to the use of shape grammars in design.

37 Since I started working with shape grammars this has been a common reaction every time I explain the concepts involved in rule-based approaches. Due to the designer's continuous quest for originality, they all want to affirm the uniqueness of their methods and design language. As such, the usual reaction is to say that they do not follow any rules, even if the rules might be of their own making.

The first is the possibility of exploring a world of potential solutions quite easily. The second is the possibility of designing systems of solutions rather than one single design solution. The latter is particularly interesting in the field of urban design because flexibility and a flexible space are part of the intended goal of an urban plan and grammar-based systems do respond to such needs. Another aspect of the problems of shape grammars is that the formal aspects of rules are not important at all to most designers, who are more interested in the meaning of particular shapes in particular contexts – semantics rather than formal composition. This is particularly significant in the case of urban design, as formal composition is far from being the most important aspect of the process. Description grammars, however, are capable of dealing with these aspects, as shown in this thesis.

The urban induction pattern formalism presented in this thesis represents a step forward in terms of the use of shape grammars in design synthesis by providing an approach that respects typical design practice. The rules are more or less hidden from the designers, who are instead presented with optional moves using recognisable names for the moves themselves and for their parameters. The design, as well as the design language is built up progressively from individual decisions in similar ways to those found in traditional practice. By using UIPs, the designer deals with concepts s/he understands and only has an indirect relationship with the rules. Meaning is guaranteed by the use of discursive grammars operating with objects found in a relational structure – an ontology – for the concepts manipulated in the urban design process.

Recently, in an eCAADe2009 keynote speech, Stiny argued that design is recursion + embedding, stressing what could be called the main properties of shape grammars. Such an argument concerns exclusively visual reasoning with shapes and excludes other meanings attached to shapes. It could be said that it is this lack of meaning that allows Stiny to include all the shapes in a design language so easily in a single shape set S. Considering the four components of shape grammars included in Stiny's definition, a set of shapes (*S*), a set of symbols (*L*), a set of shape rules (*R*) and an initial shape (*I*), this thesis proposes replacing the set S by a relational structure of shape sets such as the ontology of the urban design process shown in this thesis. Therefore a single set of labelled shapes is replaced by a relational structure of sets of labelled shapes corresponding to the object classes in the city ontology. With such a structure a relational structure of shape grammars is obtained. However, designing also involves other concepts that cannot be represented by shapes. To deal with these concepts Stiny provided an additional formalism – description grammars (1991) – to provide additional design information related to those concepts which can be used to support the generation of design solutions. These concepts are not represented by shapes but can be attributes or properties of shapes and they are also part of the ontology. In short, this thesis proposes that design systems aiming to generate meaningful designs need to be structured as compound grammars, including shape and description grammars that are able to compute concepts taken from a relational structure of concepts within a particular design domain.

Regarding urban design, the thesis also contributes new propositions, namely a conceptual model for the development of flexible urban plans and a model for integrating design-related information into the design workflow, particularly the density-based indicators that are calculated parallel to the development of design solutions.
As this thesis has shown, the proposed model contains or promotes the use of systems, types and patterns due to their ability to capture tacit knowledge accumulated from a long history of architectural experience. As demonstrated earlier, CItyMaker can deal with these concepts and seems an adequate approach to the problem of designing flexible urban plans. It is proposed that flexibility is expressed formally by an urban grammar defining the flexibility space of the plan. An instantiation of the grammar is proposed as an example of the system's application. Therefore, the thesis proposes a formal structure for designing flexible urban plans and two prototype tools to support such a practice. However contradictory the two arguments may seem – originality versus the use of systems, types and patterns – it is important to argue here that both aspects are important in design practice and any design tool should be able to accommodate both characteristics. It should be sufficient to point out that even Le Corbusier, whose work is possibly the most radical example of breaking with history, is constantly supported by historical references. Design is always a compromise between a certain degree of experimentalism and some level of established cultural knowledge. The extent to which one dominates the other depends on the skills and responsibility of the designer. In CItyMaker, both approaches are available for designers to select. In all cases, the tools proposed here improve technical awareness regardless of the design options, due to the information displayed. The tool therefore provides the means to investigate the relationship between urban types and qualitative indicators further, thus contributing to future studies on urban typology and urban morphology.
As an additional contribution, the model was prepared to interact easily with the most common analytical tools, namely geographic information systems. Analytical tasks are particularly important in urban design in comparison to architectural practice. The pre-design analysis of the context and related data, as well as the post-design analysis of the plan integrated into the context usually involve complex analytical procedures and a large amount of geo-referenced information. Geographic information systems are particularly suitable for executing such analytical tasks or supporting the use of other analytical tools such as *space syntax* or *place syntax*, or any kind of topology-based spatial analysis. Given the importance of analytical tasks in urban design, the proposed design system is structured to receive processed data from a GIS-based analysis and generate designs in a GIS compatible format involving correctly structured representations and associated data. This feature of CItyMaker is due to the relational structure of grammars provided by the ontology which enables GIS compatible representations to be generated. Moreover, in order to enhance the designer's awareness during the design process, the system was structured to inform designers about density-based indicators after each move. This addition is perhaps one of the most interesting features of the design model and it is particularly evident

in the Model B implementation. The important role of density-based indicators in the urban design and planning process was quite clearly stressed in Berghauser-Pont and Haupt (2010) and Model B follows its mathematical model accurately in calculating the indicators.

§ 8.2 Limitations

The main limitations of the CItyMaker model are related to the wide subject area and its integration within the City Induction project. The main limitations of CItyMaker are:

Regarding the theoretical model:

- The model still needs developing to enlarge the scope of its use. Although partially solved in Model B, there are no UIPs including grammars to deal with curved streets. The theoretical model would certainly be improved by the addition of new UIPs, in particular for dealing with curved streets.
- The shape grammar formalism always implies some degree of language pre-definition. However, the model could be improved if the overall structure was divided into two UIP levels: high level customisable (concept oriented) patterns and detail level patterns with a higher level of automation for production.

Regarding model A:

- Programming model A is a very time consuming task and therefore the implementation is still limited in terms of the scope of its results in comparison with the expectations created by the theoretical model.
- Reflexivity depends on a history registry (not implemented yet) and in all cases implies returning to some previous stage of the design.
- The interface is less appealing than the Model B interface. However, its integration within the AutoCAD environment clearly presents it as an extension of this platform's capabilities.
- Data is only available in the database. There is no graphic treatment of data to improve visualisation and data legibility, as in Model B.

Regarding model B:

- Although the data in the Grasshopper model is always available, no data has been exported to the City Induction database yet.
- Due to the feed-forward characteristics of Grasshopper, the model has serious limitations in terms of the use of recursive functions.
- The model becomes very slow if the geometrical features become too complicated, making the computation of large urban areas more difficult.

- Model B implies the simultaneous use of two software environments, the drawing platform and the programming platform, although this also makes Model B very interactive and legible in terms of the interaction between the geometric model and the database.

Regarding the overall research:

- The proposed models do not have enough integrated calculations or tools for topology-based assessment. Considering the extensive use of topology-based post-design evaluation in urban design, the models could be greatly improved with the introduction of some real-time calculation of topology-based indicators. This, however, implies extensive research into the subject, namely cross-referencing state-of-the-art research on these topics with parametric urban design models.

The limitations concerning the two prototypes are not particularly important in terms of the validity of the proposed methods and theoretical model. In principle, they can be overcome simply with more work. However, they would certainly profit from research focussing on interface development. Nevertheless, it should be said that Model A follows the theoretical model in detail and that this structure is particularly suitable for extending the AutoCAD Civil 3D platform into a CIM platform. Extensive development of this model would provide an efficient tool for urban design. The rule-based approach not only provides GIS compatibility and data on urban indicators, but also a tool for exploring alternative solutions. Model B approaches the theoretical concepts in a different way, taking advantage of the parametric properties of Grasshopper. This model shows a high level of usability in urban design due to its parametric flexibility, good interaction at the level of the geometrical model and user-friendly interface. However, despite its good parametric features, it has many limitations in terms of recursivity, which is the main property of shape grammars and definitely the main characteristic of the theoretical model. The lack of recursivity has been partially solved using two different approaches, as shown in Section § 7.3 , page 205. However, there is a difference in usability and interactivity in the Grasshopper models that include recursivity: the recursive behaviour is made possible due to a break imposed in the information flow allowing for feedback. This characteristic decreases the interactivity of the parametric model. In fact, Model B clearly works better without the use of recursive functions. In the end, it becomes evident that there are advantages and disadvantages to both models.
The limitations regarding the theoretical model are substantially more important. The first limitation cited above can (in theory) be overcome with more work on developing complementary UIPs to extend the scope of the tool application. However, the structure of shape grammars really creates limitations. The pre-definition of design languages has been clearly indicated from the beginning of this thesis as a limitation on shape grammars. Much of the work involved in developing the theoretical model concerned this aspect and led to the UIP concept. The problem was solved with the use of independent design moves (UIPs) which can be combined in different ways by

designers to form a customised design language. However, there are still limitations, due to certain shape grammar properties. There are essentially two difficulties that need to be solved:

1 The generation of meaningful curved urban fabrics. Shape grammars computing curved shapes use very complex mathematics and face an added difficulty in terms of shape recognition. To overcome the latter, the development of shape grammars requires the development of very abstract grammars which creates extra difficulties regarding the semantic attributes of shapes.
2 The use of more complex shapes, namely curved shapes, increases the difficulties concerning subshape recognition. However, it should be possible, with the same methodology used in this thesis, to find typical design moves using curved shapes and translate them into UIPs with reasonable accuracy without losing the required flexibility. As an example, recurrent ways can certainly be found for designing streets following the lowest slope to climb hills and therefore develop design rules for a generative pattern to design streets that adapt to topography. Another pattern could generate streets that follow the level along a hilly site. In addition, research into curved shape grammars has improved recently (Jowers and Earl, 2011) and may offer new answers to issues previously considered difficult to solve.

It is not easy to fully integrate analytical tools, which are usually specific to each analytical phase, and is even questionable. Practice should be structured in such a way that those who decide what a site needs (the programmers) are not the same as those designing the formal expression of these needs (design generation) or the ones evaluating the proposals (the evaluators). The idea is to avoid corrupting each phase with conflicting interests from another phase. However, an awareness of all phases should improve decisions, since it provides more information to support decision-making. Nowadays topological analysis of street networks is one of the most important analytical processes used in urban design, and in space syntax in particular. Due to the characteristics of the software and the analytical method, analysis is essentially a post-design method[38]. In any case, because these techniques are capable of informing designers about several qualities of the street network it could be extremely useful to have intermediate analysis during the design process. This is perhaps the main limitation of CItyMaker, although the integration of CItyMaker within the City Induction model should partially overcome this problem. Nevertheless, this is still a partial solution in the sense that the analytical procedures are not directly integrated into CItyMaker but only complement it in the post-design phase – i.e. the analysis is still a post-design analysis. However, the recursive street grid shown in the development of prototype B includes the possibility of involving a partial route structure analysis which provides some insight into the qualities of the street network.

38 It can also be used in the pre-design phase, but any analysis implies the existence of a finished design.

Although promising, this feature is not totally functional yet and needs further development.

Before ending this section it should again be made clear that the scope of this thesis involves only the design phase and does not include programme formulation or design evaluation. Firstly, a designer can design urban plans with CItyMaker by informing the inputs with any means available. Independently of the City Induction components, inputs may come from any tools that a designer considers useful for the purpose and are subject to expert judgement. Secondly, CItyMaker provides the freedom to explore the available design space and produces related data which should be subject to critical judgement. This critical judgement is an evaluation process in which the criteria defining the benchmarks should be set by the appropriate means. CItyMaker only generates the data needed for evaluation and does not constrain design exploration in any way. This characteristic allows for generic application, whereas interpreting the information is still a task reserved for the designer.

Typically, the communication of ideas and concepts in urban design is influenced by architectural forms of communication, namely through the development of 3D models for visualisation. This approach involves two dangerous aspects: firstly, it tends to make the urban dependant on the architectural image, and secondly, it imposes a definitive image on what should not have been defined yet. Additionally, for production reasons, these images tend to show a monotonous environment replicating the same architectural types *ad infinitum*. CItyMaker deliberately dispenses with the image of buildings to concentrate on the urban scale parameters, maintaining architectural image as a free design field. Therefore, the generation model was not structured to generate end result 3D images but rather to develop models that enhance the reading of design support data, namely density indicators or derived qualitative indicators. This is certainly a good structure for design tools to use during the first design steps. However, there are also setbacks to this structure. For instance, the architectural image or shape does have some influence on the formation of the urban environment and captures the attention of lay people. Normally, when shown to the public, rendered models tend to be preferred to abstract models even if the abstract models contain additional data to support the design options and have the advantage of maintaining architectural flexibility. Furthermore, there are important details in terms of the definition of urban spaces that have implications for architecture, such as the location of entrances, number of entrances or street-building permeability definitions, façade continuity, etc. These paradoxical aspects of design communication need to be managed in an ethical manner according to whom the design is being shown, but this is also, of course, an invitation to manipulation by marketing. Democracy imposes an open approach but underlines the importance of disclosing supporting information and the importance of the ethical use of information and the tools for visualising it.

§ 8.3 Potential developments and future work

It is clear from the statements in the previous section that CItyMaker still contains quite extensive opportunities for future development. Three different kinds of possible future developments can be considered, first concerning the theoretical model and then the two prototype models.
In terms of the theoretical model there are two main issues that need further development: the ontology needs further detailing and research still needs to be undertaken into the development of new UIPs. Both kinds of developments are concerned with the goal of extending the application scope of the system.
Regarding the ontology development, further work should involve the representation of property and land use, specifically focusing on how such concepts and representations are used in the urban design process. The same could apply to the representation of landscape and natural features. However, because the ontology is a communication protocol common to the City Induction research project, future developments on this level should involve the whole research team. To be more precise, the work in question should involve integrating the ontology for the urban design process with the ontology developed for programme formulation (Montenegro, Beirão, and Duarte, 2011). This work is already in progress (Montenegro, Beirão, and Duarte, 2011) but will still take time to reach a satisfactory conclusion. Given the scope of CItyMaker, some ontology concepts should be detailed, comparing the existing land use standards with common design practice and, specifically, understanding how the design synthesis process uses and transforms property and land use concepts. This means not just detailing the ontology but also developing new UIPs for the synthesis of land use operations.
There are two main kinds of design operations involving property and land use (i.e. involving zones): (1) those resulting from morphology-oriented design synthesis (2) those resulting from land use-oriented design synthesis. Design operations defining morphological synthesis have implications in terms of property definition. For instance, subdividing a site with two composition axes, e.g. a *cardus* and a *decumanus*, also defines a subdivision of the property structure into four partial subdivisions of the existing structure. This kind of operation corresponds to a type (1) property and land use oriented synthesis. However, some urban design strategies may be based on strict land use operations rather than morphological composition. This practice needs further investigation and consequently the development of UIPs for this design practice. A UIP has already been developed for operations resulting from morphology-oriented design synthesis (1), specifically an automatic operation that is used each time a composition axis is applied to divide a zone into two parts. The UIP is *PZSubdiv* (*Property Zoning Subdivision*) – see Appendix 2. Basically, every time an axis is generated, this UIP is also applied in parallel, using the generated axis to subdivide the existing zone into two parts. In the case of *AddingAxis* the pattern is applied recursively,

axis after axis. However, the equivalent operation for the generation of grids by *AddingBlockCells* still needs to be developed.
With regard to the development of new UIPs, a particular set of patterns should be further investigated and explored, specifically patterns involving bottom-up district generation. This research has already begun, as can be seen in the examples of the *InitialUUnit*, *AddingBlockTypes*, and *AddingClusters* UIPs (see Appendix 2). Some derivations were actually studied by testing trial grammars for the referred UIPs but at a certain point during the research process this was seen as diverting the research to another topic – the topic of emerging urbanism. This topic is definitely interesting and the examples from work by students found in (Beirão and Duarte, 2009) clearly show the value of the approach but the research should be developed within an appropriate framework. Specifically, it should start from a discussion on how a bottom-up approach should be framed in terms of design methods, planning methods and planning goals, classifying the concepts involved in these methods, their components and attributes. Only after such a classification can UIPs be developed objectively for this purpose, which was why this line of approach was temporarily abandoned – it requires specific research on purpose and methods. Taking this into consideration, the work concerning bottom-up generation developed so far should simply be seen as speculative, although indicative of a promising research field. Some information on the studies developed can be found in Appendix 3.
Regarding the prototype models, there are still a large number of areas for future development.

As it was stated in Chapter 7 as part of the conclusions regarding Model A, although following the theoretical model accurately, the model still needs further work to achieve all the intended goals. To summarise, the main features for future development are:

- upgrading the use of references (R_{ef}s) by assigning them to lines and polygons;
- cross-referencing references with weights and meanings (using labels) in order to establish priorities based on specific classifications of existing elements;
- developing automated approaches for the classification of design elements, namely the classification of islands considering specific distribution criteria;
- improving the use of heurists to filter the solution space better;
- automating plot subdivision;
- creating typology libraries for design reuse;
- improving the network structure generated in an accurate GIS compatible format and integrating topology-based analysis;
- developing a user-friendly data output interface;
- implementing a larger number of UIPs in order to extend the design space to a widely acceptable space.

When comparing Model A with Model B it was said that the main characteristic of Model A was its accurate implementation of the theoretical model and the main characteristic of Model B was its parametric responsiveness. Improving responsiveness in Model A would certainly improve the value of the model in terms of design, data and visual interactivity. The best approach (planned in the theoretical model but not implemented) would be to keep a record of all the moves, options and parameters involved, allowing the designer to re-generate solutions from any point in the process or simply change the parameters involved. Recording options and parameters was actually the strategy used for developing customisable UIPs. What is proposed here is, in fact, to extend the possibilities of this procedure to the overall design process. From a theoretical point of view it is important to stress that the final set of UIPs defines an urban grammar which establishes the scope of the urban system for which a particular set of parameters instantiates an illustrative plan. In this sense it can be said that Model A preserves the idea presented in the theory that the flexibility of the plan relies on the flexibility of the grammar. The argument in terms of urban design is that the final implementation can be changed as needed during the process as long as the grammar remains unchanged. The grammar guarantees that the designs generated will fall within the design space of the language defined by the grammar. Consequently the urban system defined by the grammar provides a certain formal control for the plan without fixing a definitive layout.

Model B presents a different paradigm. It does not have a rule-based structure. However, it contains most of the main elements found in the theoretical structure, namely main streets or guidelines, grids, squares, block exceptions defining public buildings or other city objects, height management and the distribution of land use. All these elements involve either numerical inputs or geometric inputs. All of them define a parametric model allowing for continuous adjustment of the design either by readjusting the position or reshaping geometries or by readjusting the numerical inputs. The system outputs density-based indicators following Berghauser-Pont and Haupt's theory (2010).

These characteristics make Model B an extremely dynamic and interactive design tool but it produces only one design solution. However, the flexibility lies in the ease with which the parametric model can be changed: the plan can be readjusted at any moment.

It can also be said that Model B is not really a design tool in the sense that it implies readjusting a code based on design patterns to every new situation. For instance, if the aim is to design a plan with two sets of two main streets with different street widths, two design patterns will be needed for main streets, two curves inserted as geometry inputs for each pattern and a different street width input assigned to each pattern. This process is nevertheless easy to manage. It may be said that it is more of a design method than a design tool.

The best aspect of Model B is that it is extremely easy to extend the functionalities related to the calculation of density-based indicators. For instance, if all the input data and *FSI* and *Network Density* calculations are already available, only three additional

equations are needed to calculate the *parking performance index*: one to calculate the *parking need*, another to calculate the *parking capacity* and a third to calculate the ratio between the two. In fact, most of the calculations traditionally carried out in urban design can be very easily implemented. One of the designers interviewed mentioned that the parking performance index referred only to public parking and we also needed a control for private parking. He was right, but considering that the model already provides all the calculations for gross floor area distributed by use, private parking is not a problem to calculate. Again a few extra equations simply need to be added and the results displayed on a panel. Although a little more complex, the same kind of reasoning applies to the calculation of daylight performance[39] .
Considering the above, Model B can be very easily extended in terms of calculating density-based indicators and such calculations can be presented for three levels of aggregation: district, fabric, and block levels.
A more fundamental task regarding Model B concerns fine-tuning the overall structure of the model, taking two things into consideration:

- accurately structuring the code into sets of design patterns (Woodbury, 2010);
- reducing the geometric operations as much as possible in order to reduce processing time.

These two improvements and some extensions are already being implemented with the collaboration of City Induction grant holder Pedro Arrobas.
Finally, it would be extremely useful to develop an analytical process within this system based on the topological characteristics of the grid. This is a very interesting approach and envisaged as very important since it may be able to offer topology-based spatial analysis during the design process. However, it involves an entire research field developing parametric-based methods for topological spatial analysis by adapting the existing theory (e.g., (Hillier, 1996), (Marshall, 2005), (Ståhle, Marcus, and Karlström, 2007)). Another possible research field that could easily start with such a tool would be further morphological studies involving the relationship between urban form and density indicators and spatial analysis, providing useful continuity with the studies begun by Berghauser-Pont and Haupt.
Testing the prototypes in real-case scenarios will generate new needs in terms of improvements. This may involve fine-tuning existing features or adding new ones and the work is already in progress. The municipality of Sintra in Portugal suggested using the CItyMaker tools and methods to address the study of alternative future developments in an industrial area within its jurisdiction. The stone transformation industry in this area had never been planned and the urban structure is now reaching chaotic proportions. The area includes three small villages with traditional architecture

39 The calculation of the daylight performance index and parking performance index is defined in *spacematrix* (Berghauser-Pont and Haupt, 2010).

and a traditional structure full of urban features that could be rehabilitated. It is the aim of the municipality to study future scenarios for the area including, as main concepts, the rehabilitation of the industrial area to create an improved and qualified industrial park, rehabilitation of the existing urban centres to enhance their qualities, and an extension of the housing capacity in the area, introducing better housing and mixed use areas. The idea of the study is to envisage different development scenarios for the area, studying possible layouts and comparing them with urban indicators which may allow enough data to be produced to evaluate the likelihood of success for each scenario. In this work, the tools will be used to produce layouts and a set of pre-defined measurements – the urban indicators already available in the tool and additional ones as needed. This work is being developed with a Masters student, Pedro Arrobas, and the outcomes will include his Master's thesis. The work will be developed during the first semester of 2012. When completed, the model is expected to have been tuned and an extended set of measuring tools added, providing additional urban indicators.

9 Conclusion

This thesis addresses the problem of designing urban plans for dealing with the complex development of cities. It follows the suggestion made by several authors to define planning strategies based on flexibility. Flexibility is defined here as the capacity of a design system and method to adapt to unexpected changes in the design context or process. This definition embraces three applications of the term flexibility: (1) the design process should be flexible; (2) the resulting design (plan) should be flexible during implementation, and (3) flexible or easily adaptable throughout its existence. The thesis proposes a design method and a model for a design tool – CityMaker – aimed at designing flexible urban plans. The model was inspired by previous work using patterns and shape grammars to design urban plans (Beirão, 2005), (Beirão and Duarte 2009). Three main concepts were used to define the proposed model: (a) systems, (b) patterns, and (c) shape grammars, as detailed below.

a System is quite a general concept involving the composition of thematic elements within a thematic context – in this case the thematic context is the design of urban plans and the elements are the urban elements from which they are composed, such as streets, squares, blocks, landmark buildings, etc. A system contains a certain number of selection rules and a definition of the relationships between elements defining the system structure. Any instantiation defines the system's variant (Habraken, 2000).

b Patterns are algorithmic structures involving the recognition of a common occurrence (in design practice) – the predicate condition – for which a common generic (design) solution is provided – the consequent. A combination of patterns defines a pattern language (Alexander et al., 1977). Patterns state when and how common design actions should be applied.

c Shape grammars are also algorithmic structures involving shape transformations to generate designs. A shape grammar defines a set of transformation rules that apply step by step starting from an initial shape to generate designs (Stiny and Gips, 1972). Combined with description grammars (Stiny, 1981) and heuristics, they form discursive grammars (Duarte, 2005). This compound structure provides a rigorous formalism that is able to recognise descriptions of a contextual situation (or occurrence) – a predicate – and provides the transformation rules and descriptions of goals for a contextualised transformation providing a common solution to the recognised design problem. They are therefore an appropriate formalism for providing generative codes for patterns.

The proposed design model uses a concept called urban induction patterns (UIPs). UIPs are patterns replicating common generic design moves used by urban designers in their design practice. UIPs are provided with parallel compound forms of discursive grammars to replicate the design moves. A particular combination of UIPs defines an urban grammar or an urban design language.

The model proposed in the thesis provides a large set of urban induction patterns encompassing the many phases of the urban design process – the main composition guidelines, grids, definition of urban units, transformations, public space management, building height and density management, street and public space detailing. Designers choose UIPs from the available set to generate their designs. A design is then progressively generated move by move (UIP by UIP) as in normal design practice, allowing for reflection between moves. The design space in CItyMaker is identified by a very generic grammar defined by all the possible combinations of all the available UIPs, their available options and parameter variability space. In addition, customisable UIPs provide the opportunity to extend the available grammar into more personalised areas of design. The design space becomes personal and unbounded.

In theoretical terms, when using CItyMaker a designer develops an urban grammar and an instantiation of it. The former defines the flexibility space of the plan and the latter an illustrative plan layout.

This thesis also proposes a methodological approach to urban design involving the use of various forms of patterns, as explained in Sections § 5.2 and § 6.2 . The method uses patterns with different levels of abstraction corresponding to the different phases in the design process (see Figure 6, page 86). It covers the entire design process, involving a participatory phase, a programming phase and a design generation phase (synthesis). Evaluation procedures are applied in every phase following any valid available methods and/or using any available tools. The participatory phase involves patterns used in a general sense, that is, the identification of recurring occurrences in the environment – fact patterns corresponding to predicates identified by participants – and the proposition of conceptual goals – concept patterns defined from standard solutions as a way of expressing a development vision. These patterns have a very high level of abstraction and represent very generic goals. The programme formulation is responsible for incorporating descriptions of the former patterns into the programme and others related to the application of regulations, quality standards, criteria for distributing facilities and other relevant particulars identified through a thorough analysis of the site context. This procedure provides detailed descriptions of the programme requirements which will be used in the design generation model to design the solutions. Programming patterns therefore represent an intermediate level of abstraction, formally defining design goals and urban induction patterns defined on a detailed level to instantiate design moves. This process should be as interactive as possible in order to maintain Lawson's structure of the design process (see Figure 5, page 76).

Two prototype implementations were presented to illustrate the implementation of the theoretical model: Model A, implemented in AutoCAD, and Model B, implemented in Rhinoceros, using the Grasshopper programming interface.
Model A represents an accurate implementation of the UIP structure presented in the theoretical model (see the pattern list in Appendix 2), but it is rather slow to implement and therefore the prototype is still quite incomplete. However, designs generated with this software follow the minimum requirements for supporting GIS interoperability: separating thematic representations and data, and further separating them into sets corresponding to different representation primitives – points, lines or polygons.
Model B is an adaptation of the theoretical model to a parametric software environment, taking advantage of the interactivity and responsiveness of the software. Instead of following the UIP structure it follows a simplified structure of thematic components organised as design patterns. However, as shown in Section § 7.3 , page 205, the patterns follow the same thematic structure defined in the theoretical model (see Table 6, page 121). During the development of this model, efforts were made to develop a user-friendly tool. The tool provides the opportunity to freely adjust the set of components and the respective parameters at any time, thereby allowing for a very interactive relationship with the designer.
Both tools allow data to be extracted from any part of the geometric model. This enables a set of functions to be defined to calculate density-based indicators. The calculations progressively inform the designer about the consequences of his/her design moves in terms of density measures and density-based indicators. Any of the indicators shown in Figure 45 and Figure 46 can be calculated for any configuration of the plan. The meaning of such measurements is always context dependent but urban designers should be able to interpret this information and react to it according to contextual features. This information is, in any case, valuable because it quite often expresses planning objectives and information that need to be presented to stakeholders or planning authorities. If not for any of this purposes, it at least informs the designer about objective characteristics of the plan which cannot be directly perceived in the plan.
The main scientific contributions of the thesis are:

1 A theoretical model for an urban design tool involving generative design capabilities and a design method for its use. The theoretical model provides a structure for developing generative software for urban design compatible with a GIS representational structure. The model includes features providing automatic calculation of density-based indicators. It also provides a flexible design platform for the production of flexible urban designs. The flexibility space is defined by a specific urban grammar which is one of the products of the design process. This feature contributes to the field of computational methods applied to urban design theory.

An ontology describing the concepts involved in the urban design process. This feature contributes to the development of knowledge bases for urban design and the systematic description of cities. It is also part of the communication protocol between the three modules of City Induction and therefore its contributions extend to the contributions of this research project.

2 A shape grammar formalism for developing urban grammars, or, in other words, a formal structure for developing grammars for urban design providing GIS interoperability. This contributes to the field of shape grammar studies.

3 A set of recommendations for developing software for urban design and city information modelling, namely in terms of how it should be structured to support GIS interoperability. This contributes to the field of computational methods applied to urban design theory.

4 A design method to enhance the quality of the information flow that supports design decisions in an urban design process. The method generates automatic calculations of density-based indicators providing more grounded information for decision-making. It contributes to the field of computational methods applied to urban design theory and practice.

5 A tool for supporting studies on the relationships between urban morphology and density. It contributes towards improving awareness of the quantitative and qualitative characteristics of urban morphologies and urban types.

The contributions made by this knowledge to design practice are likely to improve the quality of urban design, its management and response to the complexity of city dynamics. The improved data flow during the design process is also likely to improve the efficiency of participatory processes in the sense that the proposed system allows for comparison with alternative scenarios and provides data on each scenario to support decision-making. Stakeholders are therefore in a much better position to evaluate design scenarios and make decisions. From a social point of view, these features provide new grounds for opening up information to communities and stakeholders, thus fostering transparency. Communities may become better informed and objective, improving the outcomes of participatory events. The common criticism that participatory events, although more democratic are technically less reliable, may definitely be overcome.

The new approach proposed in this thesis should help provide a step forward towards the production of more sustainable cities, at least in the sense that it offers a greater capacity for designing cities that can adapt to evolving societies. In other words, this thesis may contribute towards planning more responsive cities [WS1], by providing a set of tools and methods for this purpose.

References

Acioly Jr., Claudio. 2010. 'The Informal City and the Phenomenon of Slums: The Challanges of Slum Upgrading and Slum Prevention'. In *New Towns for the 21st Century; the Planned Vs. the Unplanned City*. Michelle Provoost, INTI. SUN, Amsterdam.

Alexander, C. 1964. *Notes on the Synthesis of Form*. Harvard Univ Pr.

Alexander, C. 1979. *The Timeless Way of Building*. Oxford University Press, USA.

Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. 1977. *A Pattern Language*. Oxford Univ. Pr.

Arnstein, S. R. 1969. 'A Ladder of Citizen Participation'. *Journal of the American Institute of Planners* 35 (4): 216–224.

Ascher, F. 2001. *Les Nouveaux Principes De L'urbanisme:[la Fin Des Villes N'est Pas à L'ordre Du Jour]*. Éd. de l'Aube.

Barton, H., M. Grant, and R. Guise. 2003. *Shaping Neighbourhoods: a Guide for Health, Sustainability and Vitality*. Routledge.

Batty, M. 2005. *Cities and Complexity*. MIT Pr.

Batty, M. 2009. 'Evolving Real and Ideal City Form Using Complexity Theory: a New Paradigm for City Planning'. In *Model Town: Using Urban Simulation in New Town Planning*, 57–73. SUN. Amsterdam.

Beirão, J., and J. Duarte. 2005. 'Urban Grammars: Towards Flexible Urban Design'. In *Proc. 23rd Int. eCAADe Conf*, 491–500.

Beirão, J., J. Duarte, and R. Stouffs. 2009. 'An Urban Grammar for Praia: Towards Generic Shape Grammars for Urban Design'. In *Computation: The New Realm of Architectural Design [27th eCAADe Conference Proceedings*, 575–584. Istanbul.

Beirão, J., G. Mendes, J. Duarte, and R. Stouffs. 2010. 'Implementing a Generative Urban Design Model'. In *eCAADe 2010 Conference: Future Cities*, 265.

Beirão, J. N. 2005. 'Gramáticas Urbanas: por uma Metodologia de Desenho Urbano Flexível'. Master Thesis, Lisboa: ISCTE. http://www.bquadrado.com/paginas_web/targets/grammars.html.

Beirão, J. N, and J. P Duarte. 2009. 'Urban Design with Patterns and Shape Rules'. In *Model Town, Using Urban Simulation in New Town Planning*. Egbert Stolk and Marco te Brömmelstroet. SUN, Amsterdam.

Beirão, J. N, J. P Duarte, N. Montenegro, and J. Gil. 2009. 'Monitoring Urban Design Through Generative Design Support Tools: a Generative Grammar for Praia'. In *Proceedings of the APDR Congress*.

Beirão, J. N, J. P Duarte, and R. Stouffs. 2011. 'Creating Specific Grammars with Generic Grammars: Towards Flexible Urban Design'. *Nexus Network Journal*: 1–39.

Beirão, J., Pirouz Nourian, and Bardia Mashhoodi. 2011. 'Parametric Urban Design: An Interactive Sketching System for Shaping Neighborhoods'. In *Proceedings of the Conference eCAADe 2011*. Ljubljana.

Beirão, J., Pirouz Nourian, and Bart van Walderveen. 2011. 'Parametric "Route Structure" Generation and Analysis: An Interactive Design System Application for Urban Design'. In *IASDR 2011*. Delft.

Beirão, J., 2002. 'Baixa Pombalina'. Unpublished scholar essay.

Berghauser-Pont, B., and P. Haupt. 2010. *Spacematrix. Space, Density and Urban Form*. NAI.

Boeijenga, J., J. Mensink, and J. Grootens. 2008. *Vinex Atlas*. 010 Publishers.

Bonfiglioli, S., G.P. Calza, and S. Stabilini. 2009. 'Contemporary Cities as Changing Architectures of Time: Concepts and Instruments of Urban Time Planning - the Case of Bergamo'. In *Model Town: Using Urban Simulation in New Town Planning*, 96–117. SUN. Amsterdam.

Bosma, K., D. Van Hoogstraten, and M. Vos. 2000. *Housing for the Millions: John Habraken and the SAR (1960-2000)*. NAi Publishers.

Brandt, E. 2006. 'Designing Exploratory Design Games: a Framework for Participation in Participatory Design?' In *Proceedings of the Ninth Conference on Participatory Design: Expanding Boundaries in design-Volume 1*, 57–66.

Brandt, E., and J. Messeter. 2004. 'Facilitating Collaboration Through Design Games'. In *PDC*, 4:121–131.

Buelinckx, H. 1993. 'Wren's Language of City Church Designs: a Formal Generative Classification'. *Environment and Planning B* 20: 645–645.

Burden, D., M. Wallwork, K. Sides, R. Trias, and H. B. Rue. 2002. *Street Design Guidelines for Healthy Neighborhoods*. Center for Livable Communities.

Carmona, M., S. Marshall, and Q. Stevens. 2006. 'Design Codes: Their Use and Potential'. *PROG PLANN* 65: 207.

Celani, Gabriela, J. Beirão, J. Duarte, and Carlos Vaz. 2011. 'Optimizing the "characteristic Structure": Combining Shape Grammars and Genetic Algorithms to Generate Urban Patterns'. In *Proceedings of the Conference eCAADe 2011*. Ljubljiana.

Chomsky, N. 1957. 'Syntactic Structures (The Hague: Mouton, 1957)'. *Review of Verbal Behavior by BF Skinner, Language* 35: 26–58.

Correa, C. 2000. *Housing and Urbanisation*. Urban Design Research Institute. Thames & Hudson.

Cross, N. 2007. *Designerly Ways of Knowing*. Birkhauser.

Darke, J. 1979. 'The Primary Generator and the Design Process'. *Design Studies* 1 (1): 36–44.

Dorst, Kees. 2004. 'The Problem of Design Problems'. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.6547.

Duarte, J. P. 2001. 'Customizing Mass Housing: a Discursive Grammar for Siza's Malagueira Houses'. PhD Dissertation, MIT.

Duarte, J. P. 2005. 'A Discursive Grammar for Customizing Mass Housing: The Case of Siza's Houses at Malagueira'. *Automation in Construction* 14 (2): 265–275.

Duarte, J. P, J. N Beirão, N. Montenegro, and J. Gil. 2012. 'City Induction: Formulating, Generating, and Evaluating Urban Plans'. In *Digital Urban Modelling and Simulation*. CCIS Series Communications in Computer and Information Science Series. Springer-Verlag.

Duarte, José P, and José Beirão. 2011. 'Towards a Methodology for Flexible Urban Design: Designing with Urban Patterns and Shape Grammars'. *Environment and Planning B: Planning and Design* 38 (5): 879 – 902. doi:10.1068/b37026.

Duchamp, M. 1957. 'The Creative Act'. *Creative Art Convention of the American Federation of Arts, Houston, Texas, April, 1957*.

Fleisher, A. 1992. 'Grammatical Architecture?' *Environment and Planning B: Planning and Design* 19 (2): 221–226.

Flemming, U. 1987. 'More Than the Sum of Parts: The Grammar of Queen Anne Houses'. *Environment and Planning B: Planning and Design* 14 (3): 323–350.

Friedman, A. 1997. 'Design for Change: Flexible Planning Strategies for the 1990s and Beyond'. *Journal of Urban Design* 2 (3): 277–295.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software*. Vol. 206. Addison-wesley Reading, MA.

Gausa, M., P. Hammond, and P. Hammond. 1998. *Housing: New Alternatives, New Systems*. Birkhäuser.

Gausa, M., and J. Salazar. 2002. *Housing+ Single-family Housing*. Birkhäuser.

Gil, J., J. Beirão, N. Montenegro, and J. Duarte. 2010. 'Assessing Computational Tools for Urban Design'. In *eCAADe 2010 Conference: Future Cities*, 361.

Gil, J., and J. P Duarte. 2008. 'Towards an Urban Design Evaluation Framework'. In *Architecture in Computro- 26th eCAADe Conference Proceedings, Antwerpen*, 257–264.

Gil, Jorge, N. Montenegro, J. Beirão, and J. Duarte. 2009. 'On the Discovery of Urban Typologies: Data Mining the Multi-dimensional Character of Neighbourhoods'. http://cumincad.scix.net/cgi-bin/works/Show?_id=ecaade2009_148&sort=DEFAULT&search=Gil&hits=7.

Gips, J. 1999. 'Computer Implementation of Shape Grammars'. In *NSF/MIT Workshop on Shape Computation*.

Gips, J., and G. Stiny. 1980. 'Production Systems and Grammars: a Uniform Characterization'. *Environment and Planning B* 7 (4): 399–408.

Grazziotin, P. C, B. Turkienicz, L. Sclovsky, and C. Freitas. 2004. 'Cityzoom: A Tool for the Visualization of the Impact of Urban Regulations'. In *Proceedings of the 8th Iberoamerican Congress of Digital Graphics*, 216–220.

Gruber, T. R. 1993. 'A Translation Approach to Portable Ontology Specifications'. *Knowledge Acquisition* 5: 199–199.

Gupta, Y. P, and S. Goyal. 1989. 'Flexibility of Manufacturing Systems: Concepts and Measurements'. *European Journal of Operational Research* 43 (2): 119–135.

Den Haag. 2010. 'Bestemmingsplannen Ypenburg'. *Bestemmingsplannen Leidschenveen-Ypenburg*. http://www.denhaag.nl/home/bewoners/to/Bestemmingsplannen-LeidschenveenYpenburg.htm.

Habraken, N. J. 1972. 'Supports: An Alternative to Mass Housing'. *LONDON: ARCHITECTURAL PRESS(1972), 97 PP.(General)*.

Habraken, N. J. 1980. 'Design for Adaptability, Change and User Participation'. *Country: Indonesia*: 23–29.

Habraken, N. J. 1988. 'Type as a Social Agreement'. In *Proceedings of the Asian Congress of Architects, Seoul*.

Habraken, N. J. 2000. 'The Structure of the Ordinary'. *URBAN DESIGN QUARTERLY*: 40–40.

Habraken, N. J, J. T Boekholt, P. J. M. Dinjens, and A. P. Thijssen. 1976. 'Variations: The Systematic Design of Supports'. *MIT*.

Habraken, N. J, and M. D Gross. 1987. *Concept Design Games*. Dept. of Architecture, MIT.

Halatsch, J., A. Kunze, and G. Schmitt. 2008. 'Using Shape Grammars for Master Planning'. *Design Computing and Cognition'08*: 655–673.

Halatsch, J., A. Kunze, and G. Schmitt. 2009. 'Value Lab: A Collaborative Environment for the Planning of Future Cities'. In *Proceedings of eCAADe*. Vol. 27.

Heisserman, J. A. 1991. 'Generative Geometric Design and Boundary Solid Grammars'.

Hillier, B. 1996. *Space Is the Machine*. Citeseer.

Hillier, B., and J. Hanson. 1984. *The Social Logic of Space*. Vol. 2. Cambridge University Press Cambridge.

Jacobi, M., J. Halatsch, A. Kunze, G. Schmitt, and B. Turkienicz. 2009. 'A Grammar-based System for the Participatory Design of Urban Structures'. *Proceedings of SIGraDI*.

Jacobs, J. 1961. *The Death and Life of Great American Cities*. Vintage.

Jacobs, J. 1970. 'The Economy of Cities.' *The Economy of Cities*.

Janssen, P. 2006. 'The Role of Preconceptions in Design'. *Design Computing and Cognition'06*: 365–383.

de Jong, T. 2009. 'The Evolution of a Design, Genes, Combinations, Mutations and a Selective Environment'. Lecture in the Botanical Garden for technical plants TUDelft at 11th of September 2009.

Jowers, I., and C. Earl. 2011. 'Implementation of Curved Shape Grammars'. *Environment and Planning B: Planning and Design* 38 (4): 616–635.

Karimi, K., A. Rose, M. Martinez, and N. Raford. 2009. 'New Towns of England: Understanding Failure with Space Syntax'. In *Model Town: Using Urban Simulation in New Town Planning*, 19–45. SUN. Amsterdam.

Knight, T. 1999. 'Applications in Architectural Design, and Education and Practice'. In *NSF/MIT Workshop on Shape Computation*. Vol. 67.

Knight, T. 2003. 'Computing with Emergence'. *Environment and Planning B* 30 (1): 125–156.

Knight, T. W. 1983a. 'Transformations of Languages of Designs: Part 2'. *Environment and Planning B: Planning and Design* 10 (2): 129–154.

Knight, T. 1983b. 'Transformations of Languages of Designs: Part 3'. *Environment and Planning B: Planning and Design* 10 (2): 155–177.

Knight, T. 1993. 'Color Grammars: The Representation of Form and Color in Designs'. *Leonardo* 26 (2): 117–124.

Knight, T. 1999. 'Shape Grammars: Six Types'. *Environment and Planning B* 26: 15–32.

Kolbe, Thomas H. 'CityGML Virtual 3D City Models'. *CityGML*. http://www.citygml.org/.

König, Reinhard. 2009. 'Generating Urban Structures: New Town Planning with Cellular Automata'. In *Model Town: Using Urban Simulation in New Town Planning*, 118–134. SUN. Amsterdam.

Koning, H., and J. Eizenberg. 1981. 'The Language of the Prairie: Frank Lloyd Wright's Prairie Houses'. *Environment and Planning B* 8 (3): 295–323.

Krishnamurti, R. 1980. 'The Arithmetic of Shapes'.

Krishnamurti, R. 1981. 'SGI: a Shape Grammar Interpreter'. *Research Report, Centre for Configurational Studies, The Open University, Milton Keynes, UK*.

Krishnamurti, R. 1992a. 'The Maximal Representation of a Shape'. *Environment and Planning B: Planning and Design* 19 (3): 267–288.

Krishnamurti, R. 1992b. 'The Arithmetic of Maximal Planes'. *Environment and Planning B: Planning and Design* 19 (4): 431–464.

Krishnamurti, R., and C. F. Earl. 1992. 'Shape Recognition in Three Dimensions'. *Environment and Planning B* 19: 585–585.

Kunze, A., and G. Schmitt. 2010. 'A Conceptual Framework for the Formulation of Stakeholder Requirements'. http://cumincad.scix.net/cgi-bin/works/Show?_id=ecaade2010_167&sort=DEFAULT&search=Kunze&hits=5.

Lach, D., and P. Hixson. 1996. 'Developing Indicators to Measure Values and Costs of Public Involvement Activities'. *Interact: The Journal of Public Participation* 2 (1): 51–63.

Lawson, B. 2006. *How Designers Think: The Design Process Demystified*. Architectural press.

Lehnerer, Alex. 2009. 'The City of Kaisersrot: Not a Design, but the Result of a Mediated Process of Negotiation'. In *Model Town: Using Urban Simulation in New Town Planning*, 135–147. SUN. Amsterdam.

Li, A. I. 2001. 'A Shape Grammar for Teaching the Architectural Style of the Yingzao Fashi'.

Li, I. K., and L. M. Kuen. 2004. 'A Set-based Shape Grammar Interpreter, with Thoughts on Emergence'. In *First International Conference on Design Computing and Cognition Workshop*.

Liew, H. 2003. 'SGML: A Shape Grammar Meta-language'. In *In W Dokonal and U Hirschberg (eds), Digital Design, Proceedings of the 21st Conference on Education in Computer Aided Architectural Design in Europe*.

de Maar, Birgitte. 1999. *Een Zee Van Huisen (A Sea of Houses)*. THOTH Publishers.

Marshall, S. 2002. *A First Theoretical Approach to Classification of Arterial Streets*. ARTISTS Deliverable D.

Marshall, S. 2005. *Streets & Patterns*. Routledge.

Martin, L. 1972. 'The Grid as Generator'. In *Urban Space and Structures*, 6–27. Martin, L. and March, L. (editors). Cambridge: Cambridge University Press.

Mayer, I., G. Bekebrede, A. Bilsen, and Q. Zhou. 2009. 'Beyond SimCity: Urban Gaming and Multi-actor Systems'. *Model Town. Using Urban Simulation in New Town Planning*: 168–181.

MCGILL, M. C. 2004. 'SHAPER 2D'. In *Proceedings of DCC04*.

Minsky, M. 1988. *The Society of Mind*. Simon and Schuster.

Montenegro, N. 2010. 'Building a Pre-Design Ontology Towards a Model for Urban Programs'.

Montenegro, N., Jorge Gomes, Paulo Urbano, and J. P Duarte. 2011. '4CitySemantics: GIS-Semantic Tool for Urban Intervention Areas'. In *Proceedings of the 7th Virtual Cities and Territories Conference - 7VCT*, 549–554.

Montenegro, Nuno, J. N Beirão, and J. P Duarte. 2011. 'Public Space Patterns: Towards a CIM Standard for Urban Public Space'. In *RESPECTING FRAGILE PLACES [29th eCAADe Conference Proceedings / ISBN 978-9-4912070-1-3], University of Ljubljana, Faculty of Architecture (Slovenia) 21-24 September 2011*, 79–86. Ljubljana.

Montenegro, Nuno, J. P Duarte, Paulo Urbano, and Jorge Gomes. 2011. 'An OWL2 Land Use Ontology: LBCS'. In *Computational Science and Its Applications - ICCSA 2011 Lecture Notes in Computer Science*, Volume 6783/2011:185 - 198.

Montenegro, Nuno, Jorge Gomes, Paulo Urbano, and J. P Duarte. Forthcoming. 'A Land Use Planning Ontology: LBCS'. *Lecture Notes in Computer Science (LNCS), in Future Internet, Special Issue on 'NeoGeography and WikiPlanning', Murgante, B; Borruso, G.; Gibin, M (eds).*

Moughtin, C. 2000. *Urban Design: Method and Techniques*. Architectural Press.

Moughtin, C. 2003. *Urban Design: Street and Square*. Architectural Pr.

Moughtin, C., and P. Shirley. 2005. *Urban Design: Green Dimensions*. architectural press.

Mozas, J, and AF Per. 2002. 'Ypenburg Country Estate'. *A+t*.

Muller, P. 2006. 'Procedural Modeling of Cities'. In *ACM SIGGRAPH 2006 Courses*, 139–184.

Müller, P., P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. 2006. 'Procedural Modeling of Buildings'. *ACM Transactions on Graphics (TOG)* 25 (3): 614–623.

Neufert, E. 2006. *Arte De Proyectar En Arquitectura*.

Newell, A., and H. A Simon. 1972. *Human Problem Solving*. Vol. 104. 9. Carnagie-Mellon University: Prentice-Hall Englewood Cliffs, NJ.

Noy, N.F., M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson, and M.A. Musen. 2001. 'Creating Semantic Web Contents with Protege-2000'. *Intelligent Systems, IEEE* 16 (2): 60–71.

Pak, Burak, and Johan Verbeke. 2011. 'Usability as a Key Quality Characteristic for Developing CAAD Tools and Environments'. In *RESPECTING FRAGILE PLACES [29th eCAADe Conference Proceedings / ISBN 978-9-4912070-1-3], University of Ljubljana, Faculty of Architecture (Slovenia) 21-24 September 2011*, 269–278. Ljubljana.

Parish, Yoav, and Pascal Muller. 2001. 'Procedural Modelling of Cities'. *Proceedings of ACM SIGGRAPH 2001, ACM Press, E. Fiume, Ed.* 301–308.

Pedro, J. B. 2002a. 'Programa Habitacional, Vol. 4–Vizinhança Próxima'. *Housing Program* 4.

Pedro, J. B. 2002b. *Programa Habitacional, Vol. 4–Vizinhança Próxima,(Housing Program, Vol. 4–Close Neighborhood)*. LNEC, Lisbon.

Portugali, J. 2000. *Self-organization and the City*. Springer Verlag.

Portugali, J. 2009. 'Self-Organization, Cognition and Planning: Some Implications of New Towns and Urban Simulation Models'. In *Model Town: Using Urban Simulation in New Town Planning*, 46–56. SUN. Amsterdam.

Projectbureau IJburg. 1996. 'Bestemmingsplannen IJburg'. *Gemeente Amsterdam IJburg.nl*. http://www.ijburg.nl/main.php?obj_id=196754565.

Provoost, Michelle. 2010. *New Towns for the 21st Century; the Planned Vs. the Unplanned City*. Michelle Provoost, INTI. SUN, Amsterdam.

Prusinkiewicz, P., and A. Lindenmayer. 1991. 'The Algorithmic Beauty of Plants (The Virtual Laboratory)'.

Romão, Luís. 2005. 'SGtools: A Computer Tool for Exploring Designs with Set Grammars'. http://cumincad.scix.net/cgi-bin/works/Show?_id=cf2005_2_21_64&sort=DEFAULT&search=Rom%c3o&hits=1.

Salingaros, N. A. 2000. 'The Structure of Pattern Languages'. *Architectural Research Quarterly* 4 (02): 149–162.

Schön, D. A. 1983. *The Reflective Practitioner*. Basic books New York;

Schön, D. A. 1987. *Educating the Reflective Practitioner*. Jossey-Bass.

Schön, D. A, and G. Wiggins. 1992. 'Kinds of Seeing and Their Functions in Designing'. *Design Studies* 13 (2): 135–156.

Simon, H.A. 1981. 'The Sciences of the Artificial. 1969'. *Massachusetts Institute of Technology*.

Slocum, Nikki. 2003. *Participatory Methods Toolkit: A Practitioner's Manual*.

Stahle, A., L. Marcus, and A. Karlström. 2005. 'Place Syntax-geographic Accessibility with Axial Lines in GIS'.

Stahle, A., L. Marcus, and A. Karlström. 2007. 'Place Syntax Tool—GIS Software for Analysing Geographic Accessibility with Axial Lines'. *New Developments in Space Syntax Software*: 35–42.

Steiner, F. R, and K. Butler. 2007. *Planning and Urban Design Standards*. John Wiley and Sons, Inc.

Stiny, G. 1977. 'Ice-ray: a Note on the Generation of Chinese Lattice Designs'. *Environment and Planning B* 4 (1): 89–98.

Stiny, G. 1980. 'Introduction to Shape and Shape Grammars'. *Environment and Planning B: Planning and Design* 7 (3): 343 – 351. doi:10.1068/b070343.

Stiny, G. 1981. 'A Note on the Description of Designs'. *Environment and Planning B* 8 (3): 257–267.

Stiny, G. 1991. 'The Algebras of Design'. *Research in Engineering Design* 2 (3): 171–181.

Stiny, G. 1992. 'Weights'. *Environment and Planning B: Planning and Design* 19 (4): 413–430.

Stiny, G. 2005. *Shape: Talking About Seeing and Doing*. The MIT Press.

Stiny, G., and J. Gips. 1972. 'Shape Grammars and the Generative Specification of Painting and Sculpture'. *Information Processing* 71: 1460–1465.

Stiny, G., and L. March. 1981. 'Design Machines'. *Environment and Planning B* 8 (3): 245–255.

Stiny, G., and W. J Mitchell. 1978. 'The Palladian Grammar'. *Environment and Planning B* 5 (1): 5–18.

Stiny, G. N. 1985. 'Computing with Form and Meaning in Architecture'. *Journal of Architectural Education (1984-)* 39 (1): 7–19.

Stolk, Egbert, and Marco Brömmelstroet. 2009. *Model Town: Using Urban Simulation in New Town Planning*. SUN. Amsterdam.

Stouffs, R. 1994. 'The Algebra of Shapes'. Carnegie Mellon University.

Tan, Ekim. 2009. 'The Responsive City TReC'. *The Responsive City TReC*.

Taylor, I. A. 1959. 'The Nature of the Creative Process'. *Creativity*: 51–82.

Theunissen, K. 2009. *Nieuwe Open Ruimte in Het Woonensemble*. NAI.

Timmermans, H. 2009. 'Activity-travel Patterns as Essential Ingredients for New Town Planning and Design'. In *Model Town: Using Urban Simulation in New Town Planning*, 88–95. SUN. Amsterdam.

Trescak, T., M. Esteva, and I. Rodriguez. 2009. 'General Shape Grammar Interpreter for Intelligent Designs Generations'. In *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, 235–240.

Turkienicz, B., B. B Gonçalves, and P. Grazziotin. 2008. 'CityZoom: a Visualization Tool for the Assessment of Planning Regulations'. *International Journal of Architectural Computing* 6 (1): 79–95.

Venema, Hans. 2000. *Buitenplaats Ypenburg, Een Bevlogen Bouwlocatie*. THOTH Publishers.

Venhuizen, Hans. 2010. *Game Urbanism, Manual for Cultural Spatial Planning*. VALIZ, Amsterdam.

Witten, I. H, and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Pub.

Wonka, P. 2006. 'Procedural Modeling of Architecture'. In *ACM SIGGRAPH 2006 Courses*, 17–83.

Woodbury, R. 2010. *Elements of Parametric Design*. Routledge.

Yazar, Tugrul, and B. Çolakoğlu. 2007. 'QSHAPER'. http://cumincad.scix.net/cgi-bin/works/Show?_id=ecaade2007_110&sort=DEFAULT&search=Colakoglu&hits=6.

Websites

[WS1] http://www.theresponsivecity.org/

[WS2] http://cityinduction.fa.utl.pt/

[WS3] http://archive.unu.edu/hq/library/Collection/PDF_files/CRIS/PMT.pdf

[WS4] http://www.shapegrammar.org/biblio.html

[WS5] http://www.holisticcity.co.uk/index.php?option=com_content&task=view&id=117&Itemid=228

[WS6] http://www.procedural.com/cityengine/features/2010.html

[WS7] http://www.cityzoom.net/

[WS8] http://usa.autodesk.com/civil-3d/

[WS9] http://www.citygml.org/

[WS10] http://www.un-documents.net/ocf-02.htm

[WS11] http://www.bentley.com/en-US/Promo/Generative+Components/

[WS12] http://www.measurb.org/en/workshop.html


KIZILAY SÜRÜCÜ KURSU
HEDEF KPSS 420
BİLGİSAYAR KURSUNDA ŞOK KAMPANYA
ATİ BİLGİSAYAR KURSU
HEDEF KPSS 450
HEDEF DERSANESİ
BANKASI
ÇANKAYA BELEDİYESİ
Cumhuriyet, demokratik idarenin tam ve mükemmel bir ifadesidir. Bu rejim, halkın gelişimini ve yükselişini sağlayan, onlardan esirlik, soysuzluk, dalkavukluk hislerini uzaklaştıran bir yoldur.
Cumhuriyet fazilettir.
Çankaya Herkes İçin
YENİ TOPLUMCU BELEDİYE...
BÜLENT TANIK
ÇANKAYA BELEDİYE BAŞKANI
Her Çocuk Süt İçecek...
HİÇBİR ÇOCUK YATAĞA AÇ GİRMEYECEK!
BÜLENT TANIK
ÇANKAYA BELEDİYE BAŞKANI
Bin Çocuk Korosu

Index of Tables & Figures

Unless otherwise stated, all images, pictures, drawings and tables were produced by José Nuno Beirão.
All the pictures used in the chapter separators were photographed by José Nuno Beirão. They show various kinds of urban environments in different cities around the world. The image separator at the beginning of Chapter 3 was photographed by Irem Erbas.

Curriculum Vitae

José Nuno Beirão

General information

Date and place of birth	June, 10th, 1965, Torres Novas, Portugal.
Nationality	Portuguese
E-mail contacts:	jnb@fa.utl.pt (TU Lisbon); J.N.Beirao@tudelft.nl (TU Delft); jnb@bquadrado.com
Websites	http://www.bquadrado.com http://www.urbangrammars.com http://cityinduction.fa.utl.pt http://www.measurb.org/en/workshop.html

Degrees

Master's degree in Urban Design, Jun. 05
Department of Architecture, Instituto Superior das Ciências do Trabalho e da Empresa, ISCTE, Lisbon
Supervisor: José Pinto Duarte (IST).
Thesis: "Urban Grammars: towards a flexible urban design methodology"

Graduate Degree in Architecture, Oct. 84 – Oct. 89
Faculdade de Arquitectura, Universidade Técnica de Lisboa FAUTL
(Faculty of Architecture, Technical University of Lisbon)
Overall mark: 15 (on a scale of 0 to 20)
Final project: 16 (Supervised by João Luís Carrilho da Graça)

Professional practice in architecture and urban design

Architect and founder partner of Bquadrado arquitectos, Lda. (since Jul. 98)

Bquadrado arquitectos, Lda., Urbanismo, Arquitectura e Design
Partners: José Nuno Beirão and Miguel Salgado Braz
Portfolio at www.bquadrado.com

Worked as a practicing architect at
Gonçalo Byrne's Office;
António José Carrilho da Graça's Office;
José Lamas' Office;
João Pedro Falcão de Campos' Office.

Publications

Master's Thesis

Beirão, J. N. (2005) *Gramáticas Urbanas: por uma metodologia de desenho urbano flexível. (Urban Grammars: towards an urban design methodology).* Master's thesis in Urban Design, ISCTE, Lisbon, Portugal.

Papers in journals

Beirão, J. N., Kunze, A., Tunçer, B., Stouffs, R., and Duarte, J. P. (n.d.). Urban patterns for decision making in participatory urban design. Submitted to *Computers, Environment and Urban Systems* in December, 2011.

Montenegro, N., Beirão, J. N., and Duarte, J. P.(n.d.). Public space patterns: modelling the language of urban space, in *International Journal of Design Sciences and Technology*. Europia Productions, Paris. Accepted on October 18 2011, forthcoming.

Beirão, J. N., Duarte, J. P., Stouffs, R., and Bekkering, H. (n.d.). Designing with Urban Induction Patterns: a Methodological approach, in *Environment and Planning B: Planning and Design*. Accepted on October 14 2011, forthcoming.

Gil, J., Beirão, J. N., Montenegro, N., and Duarte, J. P. (2012). On the discovery of urban typologies: Data mining the multi-dimensional morphology of urban areas, in *Urban Morphology*. 16, 1, April, 2012, pp. 27-40.

Beirão, J. N., Duarte, J. P., and Stouffs, R. (2010). Creating generic grammars from specific grammars: towards flexible urban design. *Nexus Network Journal*, vol. 13, no. 1, April 2011, pp. 73-111.

Duarte, J. P., and Beirão, J. N. (2012). Towards a methodology for flexible urban design: designing with urban patterns and shape grammars, in *Environment and Planning B: planning and design*, vol. 38 (5) 2012, pp. 879-902.

Beirão, J., Duarte, J., Montenegro, N., and Gil, J. (2010, May). Monitoring urban design through generative design support tools: a generative grammar for praia, in M. Matalasov (Ed.) *AMIT - Architecture and Modern Information Technologies*, vol, 2 (11), May 2010.

Conference Papers

Beirão, J. N., Nourian, P., and Walderveen, B. (2011). Parametric 'Route Structure' Generation and Analysis: An interactive design system application for urban design. *Proceedings of IASDR 2011, Diversity and Unity*. Delft, The Netherlands.

Beirão, J. N., Nourian, P., and Mashoodi, B. (2011). Parametric urban design: An interactive sketching system for shaping neighbourhoods. *Proceedings of the 29th Conference on Education in Computer Aided Architectural Design in Europe – eCAADe 2011*, (pp. 225-234). Ljubljana, Slovenia.

Montenegro, N., Beirão, J. N., and Duarte, J. P. (2011). Public Space Patterns: towards a CIM standard for urban public space. *Proceedings of the 29th Conference on Education in Computer Aided Architectural Design in Europe – eCAADe 2011*, (pp. 79-86). Ljubljana, Slovenia.

Celani, G., Duarte, J. P., Beirão, J. N., and Vaz, C. (2011). Optimizing the "characteristic structure": Combining shape grammars and genetic algorithms to generate urban patterns. *Proceedings of the 29th Conference on Education in Computer Aided Architectural Design in Europe – eCAADe 2011*, (pp. 491-500). Ljubljana, Slovenia.

Montenegro, N., Beirão, J. N., and Duarte, J. P. (2011). Public Space Patterns: Modeling the language of urban space. In G. Carrara, A. Fioravanti, & A. Trento (Ed.), *Proceedings of the 13th International Conference on Advances in Design Sciences and Technology, on Connecting Brains Shaping the World. Part V – Ontology, BIM and IFC Representations*. Rome, Italy.

Gil, J., Beirão, J. N., Montenegro, N., and Duarte, J. P. (2010). Assessing Computational Tools for Urban Design: Towards a "city information model". *Proceedings of the 28th eCAADe Conference Proceedings – eCAADe 2010*, (pp. 361-369). ETH Zurich, Switzerland.

Beirão, J., Mendes, G., Duarte, J. P., and Stouffs, R. (2010). Implementing a Generative Urban Design Model: Grammar-based design patterns for urban design. *Proceedings of the 28th eCAADe Conference Proceedings*, (pp. 265-274). ETH Zurich, Switzerland.

Gil, J., Montenegro, N., Beirão, J. N., and Duarte, J. P. (2010). On the Discovery of Urban Typologies. In Pinho, P. and Oliveira, V (Eds.), *Bringing City Form Back Into Planning, Proceedings of the CITTA 3rd Annual Conference on Planning Research*, (pp. 163-176). Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.

Beirão, J. N., Montenegro, N., Gil, J., Duarte, J. P., and Stouffs, R. (2009). The city as a street system: A street description for a city ontology. *SIGraDi 2009 - Proceedings of the 13th Congress of the Iberoamerican Society of Digital Graphics*, (pp. 132-134). Sao Paulo, Brazil.

Gil, J., Montenegro, N., Beirão, J. N., and Duarte, J. P. (2009). On the Discovery of Urban Typologies: Data Mining the Multi-dimensional Character of Neighbourhoods. In B. Colakoglu, & G. Cagdas (Ed.), *Proceedings of the 27th Conference on Education in Computer Aided Architectural Design in Europe - eCAADe 2009*, (pp. 269-278). Istanbul, Turkey.

Beirão, J, Duarte, J, Gil, J, and Montenegro, N, 2009, Monitoring urban design through generative design support tools: a generative grammar for Praia, *Proceedings of APDR Congress*, Cidade da Praia, Cabo Verde.

Beirão, J, Duarte, J and Stouffs, R, 2009a, Grammars of Designs and Grammars for Designing – Grammar based patterns for urban design, *Proceedings of CAAD Futures'09 Conference*, Montreal, Canada.

Beirão, J, Duarte, J and Stouffs, R, 2009b, A Grammar for Praia: derivations on urban grid grammars, *Proceedings of the 27th eCAADe Conference*, 2009, Istanbul.

Beirão, J, Duarte, J and Stouffs, R, 2008, Structuring a Generative Model for Urban Design: Linking GIS to Shape Grammars, *Architecture in Computro - 26th eCAADe Conference Proceedings*, Antwerpen, pp. 929-938.

Beirão, JN, Duarte, JP, 2005, Urban Grammars: Towards Flexible Urban Design, in *Proceedings of the 23rd Conference on Education in Computer Aided Architectural Design in Europe, eCAADe 2005*, Lisbon, Portugal.

Book chapters

Stouffs, R., Beirão, J. N., and Duarte, J. P. (2012). Sortal Grammars for Urban Design. In S. Müller Arisona, P. Wonka, G. Aschwanden, & J. Halatsch (Eds.), *Digital Urban Modeling and Simulation. Communications in Computer and Information Science (CCIS)* (Vol. 242). Springer Berlin Heidelberg.

Duarte, J. P., Beirão, J. N., Montenegro, N., and Gil, J. (2012). City Induction: formulating, generating, and evaluating urban plans. In S. Müller Arisona, P. Wonka, G. Aschwanden, & J. Halatsch (Eds.), *Digital Urban Modeling and Simulation. Communications in Computer and Information Science (CCIS)* (Vol. 242). Springer Berlin Heidelberg.

Beirão, J. N., and Duarte, J. P. (2009). Urban Design with Patterns and Shape Rules. In E. Stolk, & M. Brömmelstroet (Eds.), *Model Town - Using Urban Simulation in New Town Planning* (ch. 9, pp. 148-165). Almere, The Netherlands: New Town Institute.

Books as editor

Duarte, J.P., and J. N. Beirão (eds.) *Customizing Mass Housing: a studio experiment*. Centro Editorial da Faculdade de Arquitectura, Universidade Técnica de Lisboa, 2003.

This publication reports on the results of the design studios conducted during the academic year 2001-2002.

Other publications on research work	Beirão J, 2011, Generative tools for flexible urban design, *Atlantis, magazine by Polis: Urban Form*, TUDelft, Delft, 22.2, August 2011, pp.39-41. http://polistudelft.nl/atlantis/22-2/
Lectures and presentations (by invitation)	Beirão, J. N. (2011). Parametric Urban design: ferramentas para projecto urbano. Seminário Métodos formais e semi-formais em Arquitectura (Seminar: Formal and semi-formal methods in Architecture). ESAP, Porto. Invited lecturer – Lunch meetings: Parametric Urban Design - Interactive tools for supporting urban design decision making, May, 12th 2011, at OTB, TUDelft. Invited lecturer – Parametric Urban Design - Interactive tools for supporting urban design decision making, April, 5th 2011, at ISCTE, Lisboa. Invited lecturer – Strengths and opportunities for shape grammar approaches to urban design / the generation module for City Induction. November, 23rd 2009, at UniCamp, Campinas, Brasil. Lecture available online at: http://www.cameraweb.unicamp.br/midia=0500001 Invited lecturer at the Faculty of Architecture, Catholic University of Viseu, Portugal. Lecture entitled "Customized Architecture and Urban Design". April, 27th, 2009. Invited lecturer at the Faculty of Architecture, TUDelft. Lecture entitled: "Customised Architectural and Urban Design". April, 12th, 2007, at TUDelft. Invited lecturer at the II International New Town Seminar, Almere, The Netherlands - 11-12, October, 2007. Lecture entitled: Urban Design with Patterns and Shape Rules. http://www.newtowninstitute.org/files/u8/summariesB.pdf
Workshop and conference organisation	Organisation of the workshop 'Generative Urban Design' for the Third Design Computing and Cognition Conference 08 in Atlanta on June 21st (workshop information and workshop proceedings available at http://mason.gmu.edu/~jgero/conferences/dcc08/). Beirão, J. N., Duarte, J. P., and Stouffs, R. (2008). *Proceedings of Workshop 1 on Generative Urban Design. 3rd International Conference on Design Computing and Cognition*. Georgia Tech, Atlanta, GA, U.S.A. Workshop South Periphery: a house. With Manuel Gausa (Inst. de Arquitectura Avanzada de Catalunya). FAUTL, 10th to 13th, March, 2005. Scientific Coordinator – José Pinto Duarte. Organizors: José Nuno Beirão, Carlos P. Sant'Ana. Workshop South Periphery: a territory. With Willy Muller (Inst. de Arquitectura Avanzada de Catalunya). FAUTL, 9th to 12th, December, 2004. Scientific Coordinator – José Pinto Duarte. Organizors: José Nuno Beirão, Carlos P. Sant'Ana.

Exhibitions

Plug-n'-play House

Exhibition 'Uma Casa Portuguesa', in Experimenta Design 2005, Lisbon Art Bienal, Lisbon, September/October 2005. Exhibition showing proposals by 12 Portuguese architect's offices for the future Portuguese house. bquadrado arquitectos proposed the Plug-n'-play House, a customisable house with an autonomous additional nucleus plugged into a rigid common space. Bquadrado was one of the 12 invited architect's offices.

September 2005

Caldas da Rainha, Urban Lab

Exhibition of student work developed at FAUTL showing the final projects by architecture graduates – supervised by José Pinto Duarte and José Nuno Beirão. Caldas da Rainha Town Hall, Caldas da Rainha, April 2004.

Commercial space architecture

Exhibition at the António Arroio School of Arts, on built commercial spaces. This exhibition focused on shoe shop projects and showed photos and detailed drawings. Invited by the School Board.

March 97

Prizes (as architect)

1990	Public Competition for a university building - ISEG. Ajuda University Campus, Lisbon. 3rd prize. In collaboration with José António Martinez, Miguel Salgado Braz and Miguel Beleza in the second phase of the competition.
1990	Public Competition – Ideas for the old wall of Óbidos (3rd prize). Honourable mention at the 1991 Évora Architecture Triennial, (jury Álvaro Siza, João Luís Carrilho da Graça, Fernando Távora, João Vieira and Mário Barradas).
1991	Limited Competition for ISCSP, Ajuda University Campus, Lisbon (4th prize). (Co-authored with architects Miguel Salgado Braz, José António Martinez and Miguel Beleza)
1993	Limited Competition for the 2nd Sintra Triennial (4th prize).
1994	Public Competition for a monument to the city of Montijo (Secil – concrete sculpture) – 2nd prize.
1998	Competition – Ideas for the Exposalão entrance hall. - 4th prize (Co-authored with architect Miguel Salgado Braz).
2004	Public Competition for a Detail Plan for Rossio – S. Gonçalo – Amarante. 1st prize. (Co-authored with architect Miguel Salgado Braz).
2004	Public Competition - Ideas for the Archaeological Area in the Ocreza Valley. 1st Honourable Mention. (Co-authored with architect Miguel Salgado Braz).

Delft University of Technology, Faculty of Architecture,
Department Architectural Engineering + Technology,
Department of Urbanism