

José Nuno Beirão

Faculty of Architecture
Delft University of Technology
Julianalaan 134
2628 BL Delft
THE NETHERLANDS
J.N.Beirao@tudelft.nl

José Pinto Duarte*

*Corresponding author

Faculdade de Arquitectura
Universidade Técnica de Lisboa
Rua Sá Nogueira
Pólo Universitário, Alto da Ajuda
1349-055 Lisbon PORTUGAL
jduarte@fa.utl.pt

Rudi Stouffs

Faculty of Architecture
Delft University of Technology
Julianalaan 134
2628 BL Delft
THE NETHERLANDS
r.m.f.stouffs@tudelft.nl

Keywords: Urban design, shape grammars, urban patterns

Research

Creating Specific Grammars with Generic Grammars: Towards Flexible Urban Design

Presented at Nexus 2010: Relationships Between Architecture and Mathematics, Porto, 13-15 June 2010.

Abstract. The aim of the City Induction project is to develop an urban design tool consisting of 3 parts: an urban programme formulation module, a generation module and an evaluation module. The generation module relies on a very generic Urban Grammar composed of several generic grammars called Urban Induction Patterns (UIPs) corresponding to typical urban design moves. Specific grammars, such as the analytical grammars inferred from our case studies, can be obtained by defining specific arrangements of Urban Induction Patterns and specific constraints on the rule parameters. We show that variations on the UIP arrangements or rule parameters can provide design variations and specific grammars to be synthesised through design exploration. It is therefore seen as a process for synthesizing a specific design grammar within the field of urban design and has two main features: (1) it allows for the synthesis of specific grammars during the design process and (2) it allows for the customization of a personal design language within the broad scope of the generic grammar.

A formal definition of Urban Grammars is presented and its application in the production of customized urban designs is demonstrated by customizing design languages using a specific compound grammar defined by a specific arrangement of generic grammars.

Introduction

1 The City Induction Project

This paper presents a detailed description of the generation module for the City Induction project. The City Induction project aims to develop an urban design tool which is defined by linking three operative modules through a common ontology, integrating knowledge structures and representations of cities.

The three modules are:

- a formulation module, which reads data on a site's context and formulates the urban programme specifications for that site (produces the urban programme) [Montenegro and Duarte 2008];
- a generation module, which generates alternative urban design solutions for the same site (produces design solutions);
- an evaluation module, which guides the generation to meet the programme's goals (guides the generation towards satisfactory designs) [Gil and Duarte 2008].

The formulation module provides its output by means of goal patterns encoded into a description grammar [Stiny 1981]. The generation module uses arrangements of generic discursive grammars [Duarte 2001] to generate urban designs, and the evaluation module uses several evaluation techniques to guide generation towards the specified requirements. In this paper we show how a set of generic discursive grammars can be used to design specific grammars for urban design.

2 Defining the problem

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 city development behaviours and all their components, and predict desirable and controllable city developments. This has been proved virtually impossible to achieve, at least by traditional means [Portugali 1999]. In addition, the constantly changing city dynamics in contemporary society has led to the growing inefficiency of the traditional layout planning approach. Flexibility and adaptability have become imperative [Archer 2001].

In order to progress towards more efficient design systems we need to develop very flexible and interactive platforms that are able to assess the complexity of urban systems without interfering with the typical indeterminate design exploration procedures that designers adopt. Previous work [Beirão and Duarte 2009] has shown that designing urban plans with shape grammars [Stiny and Gips 1972] establishes planning systems containing explicit and implicit flexibility that can be used as adaptive features in a real-world implementation where such features become extremely important. The idea is that it is possible to define design systems that establish an embedded order through a set of design rules whilst still retaining the adaptive features that can accommodate uncertainties. In practical terms, the aim is to develop an urban design platform that can shift from the rigid layout paradigm to a new concept, the concept of city information modelling (CIM), and eventually extend the term modelling to monitoring by incorporating the analysis and evaluation tools provided by the evaluation module.

However, the implementation of shape grammars contains problems of its own. The problem of shape recognition [Yue et al 2009] has been pointed out many times as its main technical restriction, whilst it is claimed that the mathematical definitions, being founded on visual reasoning, support the type of visual ambiguity found in design [Knight 2003; Stiny 2005]. Moreover, ambiguity conflicts with design control, whilst the definition of a very detailed and complete grammar conflicts with design freedom. The latter problem arises from the fact that a shape grammar, even if parametric, always embeds some kind of design language, imposing the inherent language on the designs generated. The problem that concerns us and led to this research is that design languages are the result of design synthesis and not the reverse, meaning that exploring design languages is not an aim of design, whereas the synthesis of a design language is. Exploration of the language is only an extension of the design capacities, not the purpose of design itself. As such, the main question therefore is how to define a shape grammar during the exploratory design process.

Therefore, the research questions are, firstly, how to design using shape grammars, given that a designer's language is usually synthesized together with the design process itself and, secondly, how to apply shape grammars in urban design in order to obtain a more flexible and efficient urban design process. Solving the first problem provides a

response to the second, as it forms the basis for the development of a supporting design tool. However, the second question involves complex features of the urban environment that make it much more difficult to answer. This research falls within the framework of urban design.

To address these problems the research used the following methodology. First, the main characteristics of the existing urban design and assessment tools were assessed, in order to figure out how shape grammars should be used in conjunction with these systems. A survey was also carried out of the supporting literature capable of providing guidance in specifying the aims of the design tool – i.e., what it should do and what it does not need to do. The following section presents the theoretical background to the research. In the third section, we propose the use of arrangements of generic grammars as a means of enabling specific design grammars to be developed. Section 4 shows the technical definitions of this concept, the structure of the generation module and how it works within the City Induction concept. Section 5 contextualises the research design space within the framework of four case studies, in order to simplify the prototype implementation, starting from the assumption that the natural approach in this case is to progress from simple to complex implementations. In this section we present some grammar examples extracted from the case studies to demonstrate that generic grammars correspond to urban induction patterns and that specific arrangements of generic grammars produce specific grammars. We may therefore call generic grammars designing grammars. The discussion section engages the reader in a critical review of the achievements of the research and the scope for future work, establishing some new hypotheses for future research. The final section draws conclusions on the achievements of the research.

3 Background and theoretical support

Architecture, urban design and urban planning are three different scales of design activity that merge within the context of the city. It is already established in literature on the subject that these scales range from local to global (or vice versa) alongside several complex interactive systems, namely social, economic, environmental and political, all of which contribute towards generating uncertainty and complexity in cities and their development [Archer 2001; Batty 2005; Portugali 1999]. In this environment, urban design becomes a rather difficult and unpredictable task.

In the current state of the art, it is impossible to find fully integrated tools that enable us to assess the many aspects relating to the task of urban design. There are several tools for urban analysis, tools for evaluation and tools for designing but no single tool seems adequate to assess all the demands of urban design. The basic distinction that is important to this paper is the distinction between GIS and CAD tools. The former are extremely powerful assessment tools for evaluating urban data and performing many analytical tasks which may inform urban design, but they are not design tools. GIS interfaces share the characteristic of gathering geo-referenced information and representations of existing components or concept-components¹ in our environment. Data and shape-files (i.e., representations) are linked by a geographical reference. GIS platforms provide many different tools that allow us to run several types of analysis. There are also other types of software or plug-ins that add other analytical functions to these platforms, space syntax [Hillier and Hanson 1989; Hillier 1998] being probably one of the most widely-used tools of this kind. Although some of these types of software

might contain some editing tools, none of them are drawing or modelling tools and for this reason, GIS platforms are very unfriendly tools for design purposes.

On the other hand, CAD tools are essentially drawing or modelling tools that do not assess data, nor do they allow for the complete topological integration of representations and data. However, most of the CAD software is already very powerful and versatile in terms of design purposes, although communication between the different platforms is difficult and implies loss of data. Nevertheless, it is clear that urban design methodologies are strongly supported by intensive analytical methods during the pre-design phases, therefore indicating the enormous potential and desirability of linking GIS and CAD to allow for analysis-design-analysis data flow cycles. Establishing the foundations of a tool for this purpose is one of the main goals of City Induction. On an urban scale, this is actually a direct translation of what Schön [1983] would call a see-move-see cycle in architectural design.

A shape grammar [Stiny and Gips 1972] is a set of shape transformation rules that are applied recursively to generate a set of designs. These are $\alpha \rightarrow \beta$ type rules in which α and β are labelled shapes from a finite set of shapes S and a finite 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 [Stiny 1980]. As Stiny has pointed out, shape 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 the space of 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 $g(\alpha) \rightarrow g(\beta)$.

Shape grammars have successively demonstrated a capacity to encode the design rules embedded in design languages with a rigorous technical formalism. However, the semantic discourse in urban design is not only provided by shape transformations but also political, social and territorial contexts which are informed by features other rather than those of form. To solve this semantic problem previously pointed out by Fleisher [1992], Duarte [2001] proposed the concept of discursive grammars, a combination of description grammars [Stiny 1981] and shape grammars, as a way of providing descriptions of designs that are appropriate for a particular context.

Previous work using shape grammars and patterns [Alexander et al 1977] as an approach to solving urban design problems has been carried out in recent years in design studios at the Technical University of Lisbon [Beirão and Duarte 2009]. This work still remains the main motivation for the City Induction research project, since it has demonstrated the potential of using shape grammars in urban design. Although the design studio was run with current tools, shape grammars were applied informally using current CAD functionalities and the analytical tasks were performed without the support of GIS-based tools, the idea of integrating analysis, generation and evaluation into a single working platform has been our main focus since then.

4 *Designing grammars for urban design*

As previously mentioned, the implementation of shape grammars implies specific technical problems. Basically, two main problems concerning shape grammars are addressed in this research: firstly, the shape recognition problem and secondly, the problem of defining the grammar during the design process. The linking of GIS and CAD representations into a compatible format defines the third problem under investigation.

Our current work focuses on the implementation of the generation module for City Induction. It proposes two devices as a means of solving the three said problems. The first problem is partially solved by the introduction of a City ontology. The third problem is entirely solved by the same device. The second problem is solved through the introduction of Urban Induction Patterns (UIPs), small generic grammars encoding urban design moves.

The City ontology defines and organizes significant relationships between the various types of objects or components found in the urban space that will be used in the urban design process. It is structured into object classes, each containing object types and attributes. At top level the City ontology contains 5 different classes – networks, blocks, zones, landscape and focal points. The ontology was defined to support communication between the three modules of City Induction, but also provides a structure that can create layered representations of city features. This layered structure is envisaged as a means of establishing the direct export of design generations to a GIS platform. It is also seen as a way of structuring urban grammars into parallel generic grammars, basing the definition of the shape sets of these grammars on the object classes of the ontology. Details on the ontology can be found in a recent paper [Beirão, Montenegro et al 2009].

In this paper we show how design moves [Schön 1983] can be encoded into small generic grammars and combined in different ways to form customized designing grammars that can therefore be used to synthesize a personal design language during the design process. Such a system may be used to improve design procedures and design exploration, as it enables the advantages of the generative properties of shape grammars to be used, whilst also allowing a personal design language to be explored in the design.

Encoding design moves into generic grammars: Urban Induction Patterns (UIP)

Donald Schön says that designs evolve through a series of see-move-see cycles. It is a reflective process that is performed continuously throughout the design process by the designer until s/he comes up with a proposed final solution. The design rules are the result of such a process. Only when the process is considered finished, is the designer able to talk about the design process and replicate the procedure. Only then, is the design capable of providing a specific grammar, a set of shape rules that can translate the design language of the architect, i.e., a consistent design expression translated into a shape grammar.

What this research intends to develop is a way to simultaneously provide the exploratory design process whilst also developing a consistent design grammar encoded into the algebraic formalism of shape grammars, so that by the end of the design process the designer can obtain a complete customized shape grammar that enables him to further explore the design space defined by the grammar. In the case of urban design, what this process provides is the possibility of designing, not a final layout, but a set of

rules that can produce any layout within the design space defined by the grammar. A design language is therefore provided without enforcing a specific layout.

The main idea underlying this work is that we can “break down” the complete design process into a particular arrangement of independent design moves. Each design move is encoded into a very generic shape grammar independent of context that can be applied to different contexts and customized by constraining the available parameters.

Generic grammars are very simple, customizable, context-independent shape grammars corresponding to generic design moves. They can be defined because what is encoded is not a complete design sequence but short recurrent design moves common to most designers. Expressions such as “defining the main axis”, “placing a landmark” or “setting the grid” are understood by any urban designer, although each individual might have a specific interpretation of their meaning. These design moves are set as design patterns [Gamma et al 1995], defining a short and very generic piece of code that generates this specific design move. Specific designs are the result of composing an arrangement of such design moves and setting specific values for the available parameters. A specific grammar is therefore the result of setting a specific arrangement of generic grammars and constraining the available parameters to the ones that express the designer’s thought language. In order to simplify the text which follows, a generic grammar encoding a recurrent urban design move will be called an Urban Induction Pattern or simply UIP.

Generic grammars for designing Urban Grammars

Another important aspect to consider is that design moves are a designer’s response to specific stimuli found in the design context. In other words, certain elements found in the context provide referential support for design decisions. These elements or references are the ones that provide the initial shapes to which a generic grammar can be applied. The references are elements that the designer selects in the design context or referential elements generated by previously applied grammars. There are therefore two ways of providing referential elements or initial shapes in order to apply a UIP: firstly, the designer selects elements in the design context and gives them the attribute R_{ef} (for referential element); alternatively, a previous grammar generates an element that is then used by another generic grammar as an initial shape.

The City Induction generation module is a design generation system based on an extremely generic Urban Grammar defined by the available UIPs and rule parameters. A specific Urban Grammar is customized through the selection of a specific arrangement of UIPs and specific constraints on the rule parameters. A specific design is instantiated for a specific selection of references (R_{ef} elements) by instantiating specific values within the constraints defined for the rule parameters. In terms of the design decision process, a specific design is obtained through three kinds of design decisions: selecting or defining references in the design context, selecting a specific arrangement of UIPs, and constraining the available parameters within the pattern rules.

These options depend on the decision taken by the formulation module or the designer. Therefore, the design process is a reflective process, responds to contextual features including regulations and quality standards, and is rule-based.

The various meanings of the term “pattern”

A brief digression is necessary here in order to explain and identify the meaning of the word “pattern” in Urban Induction Patterns. The basis of the word “pattern” as used here was first coined by Alexander et al in the book *A Pattern Language* [1977] and was defined as a recurrent problem occurring in the environment that can be clearly identified and provided with a generic solution. This concept was later extended by Gamma et al [1995] so that it could be applied in object oriented programming. The underlying idea was to identify recurrent problems in object oriented programming and to provide a detailed solution with a sample code for generic application. This extended concept was called “design pattern”, although the word “design” here means software design. An Urban Induction Pattern is a compound version of both concepts applied to urban design, i.e., an UIP is a recurrent urban design move provided with a generic grammar that replicates this design move and can be applied in many different contexts. In every sense in this context these are design patterns.

The next sub-section details this definition in terms of grammar formalisms.

Urban Induction Patterns and Urban Grammars: definitions

The top level of the City ontology defines 5 object classes, namely Networks, Blocks, Zones, Landscapes and Focal Points (fig. 1). Each object class is a set of object types divided into two major subsets: geometry (shapes, parameterized shapes) and attributes (labels). The ontology provides a dependency structure for all the shape and label sets that comprise an urban plan, allowing grammars to accept them as parameterized shapes and labels for the application of rules used to generate urban designs.

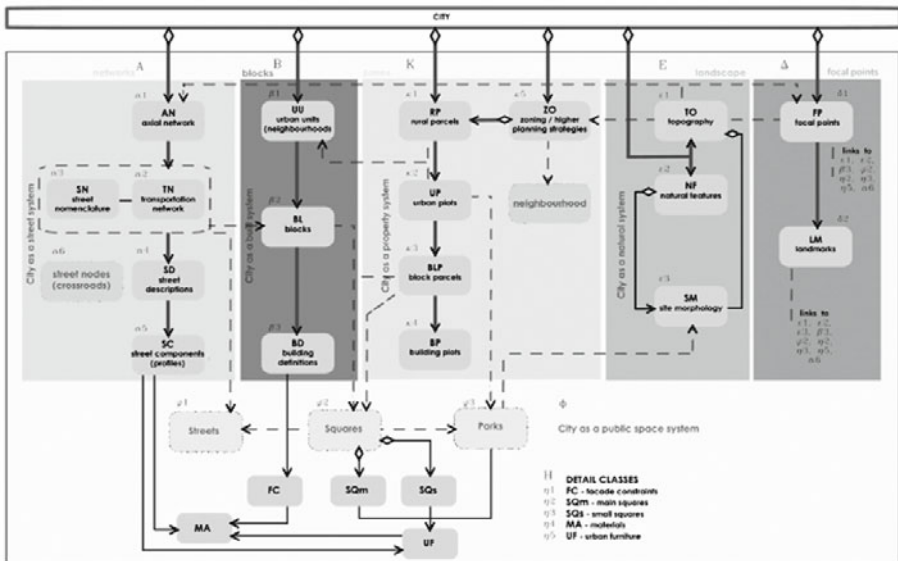


Fig. 1. City ontology – schematic approach

To simplify the notation, each set of parameterized shapes in the ontology is annotated as S_i where the index i defines the position of the set in the ontology from 1, 2, ..., n where n is the total number of shape sets in the ontology.

As defined above, an Urban Induction Pattern is a recurrent urban design move provided with a generic grammar that replicates this design move and can be applied independently of context. An Urban Grammar is a specific arrangement of generic grammars encoding urban design moves, i.e., it is an arrangement of UIPs.

The City Induction generation module was conceived of as an urban design tool or urban design system. In order to generate designs it contains a very generic urban grammar Γ which is the set of all urban grammars Γ' that can be defined using the system. Although our model is a simplified version of a possible theoretical universal model restricted to the domain defined by four case studies (see section 5), it can, in principle, be extended by (1) extending the ontology, and (2) extending the set of available UIPs.

An Urban Grammar Γ' is a Cartesian product of parallel grammars $\Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \dots \times \Gamma_n$ that use a set of parameterized shapes from the City ontology, $S_1, S_2, S_3, \dots, S_n$ respectively, to design a layer of an urban plan. Each design phase outputs a sub-design composed of several layers or, more accurately, each design phase uses some of the grammars, Γ_1 to Γ_m , in the urban grammar Γ' to generate the various layers that define the sub-design produced in that design phase. Label sets $L_1, L_2, L_3, \dots, L_n$, are the label sets in the corresponding grammars $\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_n$, and also correspond to the attribute classes in the ontology.

Generation begins with the definition of the existing elements to be used in the design. E_0 is the set of existing representations (shapes) of the working context and contains all the existing representations (shapes) regardless of the many-layered structure it may have. I_0 is the set of initial features (shapes and labels) that will be used to start the design and contains only I_s and R_{ef} objects, that is, the shape I_s , a closed polyline that represents the intervention site limit, the labels I_s and R_{ef} , and R_{ef} shapes, which are the shapes representing the selected elements in the site and context that will be used as referential guiding elements to support design rules. R_{ef} labels are attributes of the R_{ef} shapes that were selected by the designer to guide a certain stage of the design. R_{ef} shapes in fact represent the elements that guide the design decisions.

A grammar Γ' is built up from a sequence of UIPs (a sequence of design decisions) which in the end will reflect the design language of the urban plan. The same urban grammar Γ' can be used to generate different alternative designs by running the same UIP sequence again to produce different instantiations. This allows design implementations to be explored or monitored [Beirao, Duarte, Montenegro, Gil 2009]. The whole concept is, in fact, close to what could be a real algorithmic implementation of a complete Pattern Language as conceived of by Alexander [1977], but in such a way as to allow the designer to define his own pattern language.

A UIP is therefore a sub-grammar of Γ' . A UIP uses only some of the parallel grammars in Γ' , a subset Γ'' of Γ' , namely some components of $\{\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_n\}$. Each grammar Γ_i follows the definition of a discursive grammar [Duarte 2001]. Each UIP is a compound grammar Γ'' composed of a set of parallel discursive grammars Γ_i of the form $\Gamma_i = \{D, U, G, H, S_i, L_i, W, R, F, I_i\}$ where S_i is the set of parameterized 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. Each UIP addresses a goal G to be achieved through 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. W is a set of weights and F a set of functions used to constrain generation so that it respects regulations and quality standards.

5 Generating designs with Urban Induction Patterns

In this section we intend 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, we used the following approach. The work was framed by capturing UIPs only within the design space defined by four case studies, attempting to define them in such generic terms that each UIP could be used as broadly as possible regardless of context. We also tried to break down the UIP into the smallest design moves possible so that most of the large design moves would already be a composition of smaller UIPs. In fact, the analysis of the case studies demonstrated that although all the grids were quite different in terms of final results, most of them were actually obtained through different arrangements of minimum UIPs designed for this purpose.

As a general approach, this section presents the four case studies, 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 urban plans, and (2) to generate alternative solutions by applying different instantiations of rule parameters.

The four case studies

This work began with the assumption that urban designs can be broken down into very small generic design procedures or design moves which, when combined in different ways, can be used to define different design languages. In order to define such design moves in the form of UIPs as defined in the previous section, we used four urban plans as case studies.

The four urban plans are: 1) Extension plan for Cidade da Praia in Cabo Verde by Chuva Gomes; 2) Qta da Fonte da Prata (QFP), in Moita, Portugal by Chuva Gomes; 3) Ijburg/Haveneiland by Frits van Dongen, Felix Claus and Ton Schaap from a larger plan by Palmhout; and 4) Ypenburg, also by Palmhout (Palmboom and van den Bout) (fig. 2). The case studies were used to frame the work within the range of their design space. Furthermore, we assumed that to produce complete urban designs we would have to define at least four sets of rules (UIPs) 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;
- D. rules for designing details, such as the detailed design of street profiles and materiality [Beirão and Duarte 2009].

This paper focuses on the level (B) rules.

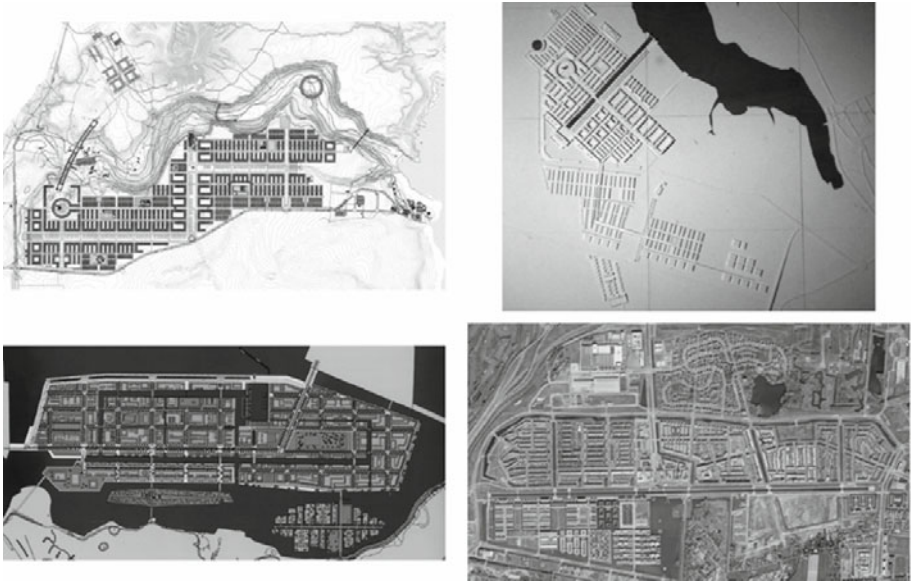


Fig. 2. Plans for (1) Praia, (2) Quinta da Fonte da Prata (both by Chuva Gomes), (3) Ijburg (by van Dongen, Claus and Schaap from a master plan by Palmhout), (4) Ypenburg (by Palmhout)

As a basic methodology we started by analysing the first case study, inferring the design rules used by the architect and defining them as Urban Induction Patterns. In interviews he has given, Chuva Gomes clearly states his design moves for both plans. He even states which moves were common and which were different in the two plans. We tried to formulate basic definitions of the UIPs in such a way that they would be valid for the four case studies or even other well known paradigmatic urban plans. Due to its simplicity, Case Study 1 provides very basic rules for designing an orthogonal grid-based plan. We tried to devise them so that, in order to obtain results like the ones in Case Study 2 the same rules (UIPs) could simply be applied with a different sequential arrangement and different values for the parameter variables. Roughly speaking, it can easily be seen that Plan 2 uses similar rules to those in Plan 1 but applies them several times in four different areas with four different orientations. Introducing more complexity into plans is therefore the result of combining basic UIPs in more complex ways. However, the two Dutch plans introduce new features suitable for encoding into other UIPs. For example, Plan 3 introduces a lot of variety into the definition of the urban block and a set of additional rules can therefore be defined from this case study. Case Study 4 introduces an even more complex variety of design transformations in comparison to the previous, less complex, case study. In this instance, not only can more variety 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.

As a general approach, considering that our purpose is the definition of a design tool, the line of reasoning always followed was that rules to explain the simpler case study would be defined first and valid variations of these rules would then be explored as a means of obtaining the design exploration potential. This task proved hard, since it was possible to lose direction when considering valid variations simply due to the fact that design possibilities are, in principle, infinite. To avoid this, we tried to remain focused

within the scope of the case studies, aiming only to prove that: (1) a minimum set of UIPs can be used to generate the four case studies; (2) the same UIPs can produce new designs in different contexts with similar rules in any language that can be composed from the UIPs available or, in other words, within the design space defined by the case studies. The additional capacities of the defined set of UIPs outside this design space were considered only as additional qualities to support the concept, but were not established as a research goal.

Defining UIPs

All the Urban Induction Patterns defined in this paper refer only to the generation of designs of urban plans in typical plan representations. 3D representations were left for future research developments and are considered here as consequence of the design generation results that might be obtained by simple extrusion of the layouts. Aspects related to topography are not detailed here, only clarified in the discussion.

As previously stated, the first UIPs applied are the ones that take the intervention site limit I_s and elements selected as references, R_{ef} as their initial shapes. The first UIP within the framework of our case studies was suggested by the explanation given by Frits Palmboom, the author of the Ypenburg plan, concerning his design methods. Regarding his first move in the design of an urban plan, in a 2008 interview made in the context of this present research, he said, “I always look for the longer line in the territory”. Taking this sentence as our motto, we called the first UIP *MainAxisisTheLongerLine*. The algorithm generates all the lines that can be generated after considering the referential elements 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 that has the closest north-south orientation. The term *cardus* already demonstrates the possibility of applications in a wider design space than the one defined by the case studies, as it is a well-known feature of classic urban planning. Following the first UIP, we devised *OrthogonalAxis*, or *Decumanus* if perpendicular to *Cardus*, as another typical design move. In Plan 2 it can be seen that these two sequential UIPs are used differently in four different areas of the site. 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). A detailed description of the rules for *Cardus* and *Decumanus* is shown in Beirão et al [2009a].

At the second design level (B) we basically considered two Urban Induction Patterns to generate the grids and a few others 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* [Beirão, Duarte, Stouffs 2009b] or by *AddingBlockCells* [Beirão, Duarte, Montenegro, Gil 2009] and have been used to reproduce the design of Plan 1. The rules for each UIP are reproduced here to facilitate comprehension of this paper. However, some rules were added to the later UIP in order to deal with the use of varying parameters. During the derivation with *AddingBlockCells*, third hierarchy axes were adjusted using *AdjustingAxistoCells*. At the end of the grid derivation, two more UIPs were used – *AddBlocktoCells* and *AdjustingBlockCells* – to create the block structures and adjust to the boundary conditions of the design respectively.

As the architect Chuva Gomes used constant parameters to define the dimensions of the blocks in the Praia plan, the results of applying any of the two UIPs are the same. However, if the parameters were to vary, for instance, within a fixed interval, the results

obtained from applying each of the algorithms would differ. Considering the goal of defining City Induction as a design tool, we view this difference as an extremely positive quality of the two UIPs, which can be used for design exploration.

Applying UIPs to generate designs

Exploring design possibilities is a task based on freely sketching many different ways of applying basic design moves to selected references within a given context. Both references and 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 solutions, even many optimum solutions, and the design exploration simply needs to find a way towards a solution space. Bearing this in mind, our aim is not necessarily to demonstrate the tool's capacity to find optimum solutions, although this may be achieved later when the three City Induction modules are connected, but to demonstrate the versatility of the design tool in exploring different design formalizations. In this sense, it is more important to show that, apart from being able to reproduce Chuva Gomes' plan, the same UIPs that were previously developed 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 references R_{ef} ; 2) applying different sequences of UIPs; 3) assigning different values to the parameters.

Although some automated ways of developing suggestions for defining references can be found, until now this first step has simply been considered a manual one. 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 designers use when sketching ideas in pencil or using any 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 where 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 well as different interpretations of the options will obviously produce different results.

Three initial UIPs are referred to in this paper – *MainAxisistheLongerLine*, *CompositionalAxis* and *Cardus*. Initial UIPs are those able to recognize the available initial shapes, which are the R_{ef} elements and the intervention site limit I_s . The basic algorithm for these three UIPs is 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. From the whole set of axes only those 10% shorter than the longer axial length will be used in the next generation steps. However, this percentage is a variable that can be manipulated by the designer by changing the amount of proposed axes. *MainAxisistheLongerLine* selects the longest available axis from the set of proposed axes. *Cardus* selects the one closest to a north-south direction. *CompositionalAxis* randomly selects one of the available axes.² References, R_{ef} labels, used by the selected axes are erased so that a second coinciding axis cannot be generated. *MainAxisistheLongerLine* and *Cardus* can be applied only once. *CompositionalAxis* can be applied as long as there are references to be used.

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.

The first level representations generated by these UIPs are axial representations of streets belonging to the AN (Network) object class in the ontology and they basically represent four 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., three times.

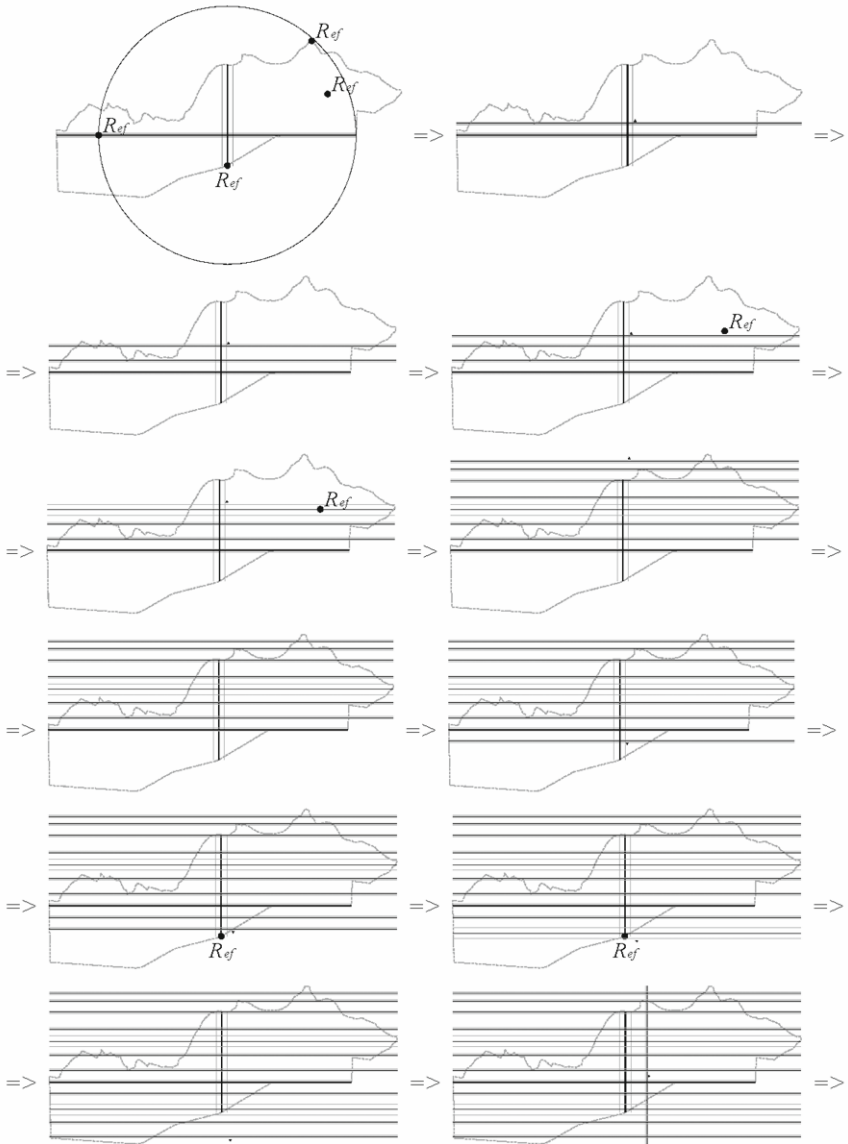
The UIPs *AddingBlockCells* and *AddingAxes* can be applied as soon as there are two orthogonal axes in the design. The parameters h and w correspond to the length and width of the urban block respectively. These parameters can be set as a fixed value, as Chuva Gomes does in Praia ($h=80m$ and $w=50m$), 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.

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 fill 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 parameters must be defined, to be used by the rules 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, while 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.

We will look first at *AddingAxes*, as this is an easier example. [Beirão, Duarte, Stouffs 2009b] show the rules and the derivation for generating the Praia plan. The rules are reproduced here (fig. 4), showing a different derivation (fig. 3). The original derivation applies an exact sequence of UIPs and a fixed value for the block parameters, ($h = 80m$ and $w = 50m$), to reproduce the design of Praia plan – Plan 1. However, if we consider an admissible range for the parameters h and w , such as the ones suggested above, variations will start to appear in the grid whilst maintaining the typical orthogonal grid appearance and street continuity. The derivation in fig. 3 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. At the end of the generation sequence, squares are applied following different algorithms for the generation of public space. The last Urban Induction Patterns apply two different building typologies to the block. These rules are not explained but their application is shown because it improves the legibility of the resulting urban plan.

Fig. 3 (this page and facing page). Derivation of the plan for Praia using AddingAxes. The derivation is simplified to the essential steps



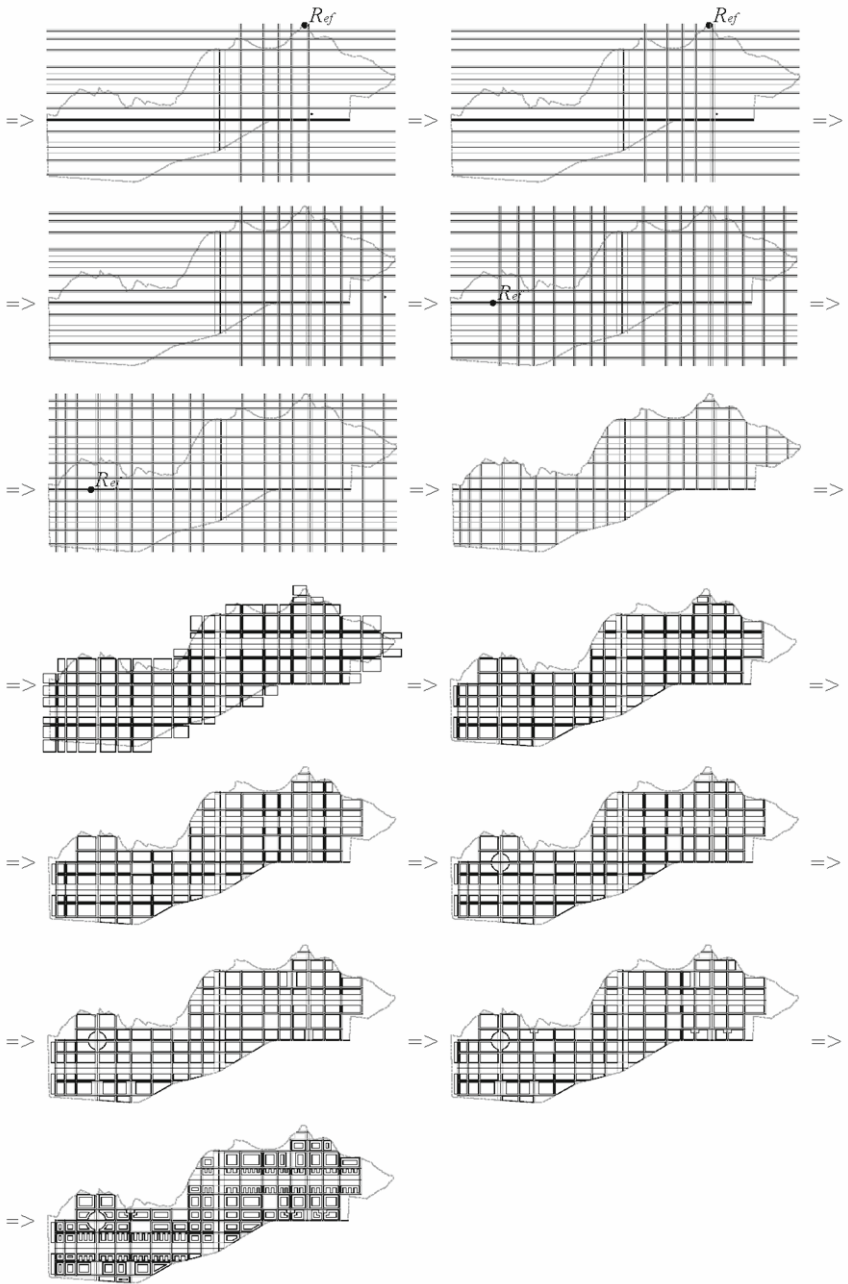


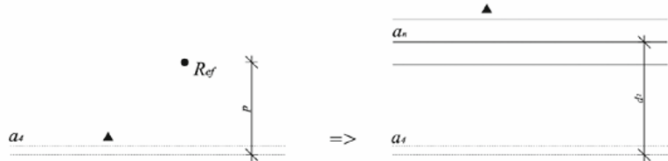


Fig. 4 (this page and facing page). Rules for AddingAxes (Rules 1, 2 and 7 are omitted)

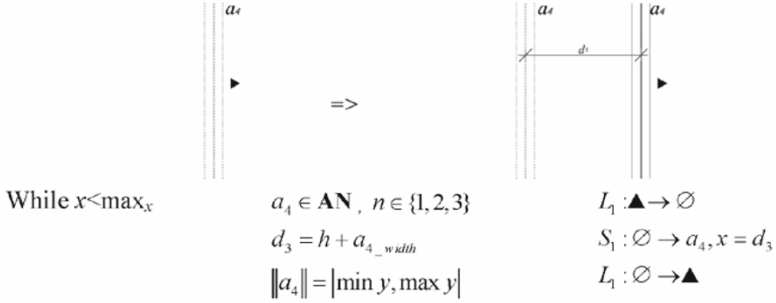
<p>UIP – <i>AddingAxes</i> – rules</p> <p>Note: the dotted shapes in the rule representations are used only for reference.</p> <p>No transformations are applied to the dotted shapes.</p> <p>Parameters h and w are block length and block width respectively.</p> <p>x and y are coordinates.</p> <p><i>AddingAxes</i>: $\Gamma = \{\Gamma_1, \Gamma_0\}$</p> $\{S_1, L_1\} \subset \Gamma_1, S_1, L_1 \in \mathbf{AN}$ $\{S_0, L_0\} \subset \Gamma_0, S_0, L_0 \in E_0$	
<p>Rule 3a</p> 	$\exists a_n, a_n' \in \mathbf{AN} \wedge a_n \perp a_n'$ $n \in \{1, 2, 3\}$ $a_4 \in \mathbf{AN}$ $d_1 = w + a_{4_width} / 2 + a_{n_width} / 2$ $\ a_4\ = \min x, \max x $ $S_1 : \emptyset \rightarrow a_4, y = d_2$ $L_1 : \emptyset \rightarrow \blacktriangle$
<p>Rule 4a</p> 	$\text{While } y < \max_y,$ $a_4 \in \mathbf{AN}$ $d_2 = w + a_{4_width}$ $\ a_4\ = \min x, \max x $ $L_1 : \blacktriangle \rightarrow \emptyset$ $S_1 : \emptyset \rightarrow a_4, y = d_2$ $L_1 : \emptyset \rightarrow \blacktriangle$
<p>Rule 4c</p> 	If: $\frac{1}{2}w - a_{4_width} < p < \frac{3}{2}w + a_{4_width}$ $\text{While } y < \max_y,$ $a_4, a_n \in \mathbf{AN},$ $n \in \{1, 2, 3\}$ $d_2 = a_{4_width} / 2 + w + a_{n_width} / 2$ $\ a_n\ = \ a_4\ = \min x, \max x $ $L_1 : \blacktriangle \rightarrow \emptyset$ $S_1 : \emptyset \rightarrow a_n, y = d_2$ $L_0 : R_{ref} \rightarrow \emptyset$ $L_1 : \emptyset \rightarrow \blacktriangle$

Rule 5a



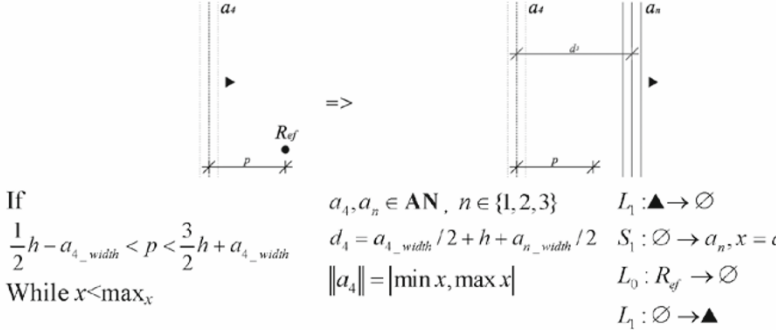
$\exists a_n, a_n' \in \mathbf{AN} \wedge a_n \perp a_n'$ $a_4, a_n, a_n' \in \mathbf{AN}, n \in \{1, 2, 3\}$ $S_1: \emptyset \rightarrow a_4, x = d_3$
 $n \in \{1, 2, 3\}$ $d_3 = h + a_{4_width} / 2 + a_{n_width} / 2$ $L_1: \emptyset \rightarrow \blacktriangle$
 $\|a_4\| = |\min y, \max y|$

Rule 6a



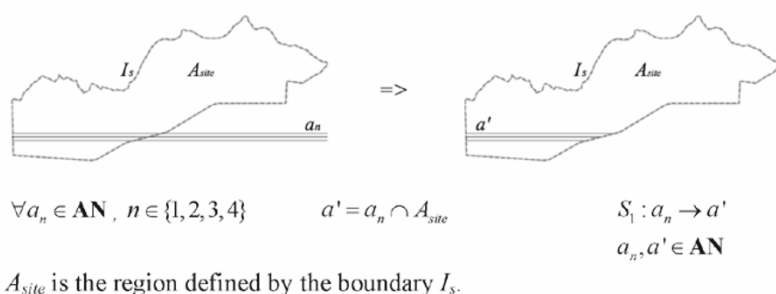
While $x < \max_x$ $a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$ $L_1: \blacktriangle \rightarrow \emptyset$
 $d_3 = h + a_{4_width}$ $S_1: \emptyset \rightarrow a_4, x = d_3$
 $\|a_4\| = |\min y, \max y|$ $L_1: \emptyset \rightarrow \blacktriangle$

Rule 6c



If $\frac{1}{2}h - a_{4_width} < p < \frac{3}{2}h + a_{4_width}$ $a_4, a_n \in \mathbf{AN}, n \in \{1, 2, 3\}$ $L_1: \blacktriangle \rightarrow \emptyset$
 $d_4 = a_{4_width} / 2 + h + a_{n_width} / 2$ $S_1: \emptyset \rightarrow a_n, x = d_3$
 While $x < \max_x$ $\|a_4\| = |\min x, \max x|$ $L_0: R_{qf} \rightarrow \emptyset$
 $L_1: \emptyset \rightarrow \blacktriangle$

Rule 8



$\forall a_n \in \mathbf{AN}, n \in \{1, 2, 3, 4\}$ $a' = a_n \cap A_{site}$ $S_1: a_n \rightarrow a'$
 $a_n, a' \in \mathbf{AN}$
 A_{site} is the region defined by the boundary I_{site} .

Apart from the fact that we let the rules 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 in applying fixed or variable parameters to the block size. The derivation in fig. 3 was simplified to the essential steps.

AddingAxes is a sub-grammar Γ'' of the generic urban grammar Γ' that we are using. Γ'' is composed of two parallel grammars, Γ_1 and $\Gamma_0 - \{\Gamma_1, \Gamma_0\}$ – 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 .

The first step in the derivation already demonstrates the result 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 show the steps where 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 on a crossroad (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).

Fig. 4 shows the UIP *AddingAxes* with the 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 not shown in this paper but they can be found in an incomplete format in [Beirão, Duarte, Montenegro, Gil 2009].

Rule 1 of the UIP *AddingAxes* maps a temporary coordinate system, $x0y$, 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:

$$\begin{aligned} \max_x &= \text{maximum } x \text{ value of } I_s \text{ in } x0y; \\ \max_y &= \text{maximum } y \text{ value of } I_s \text{ in } x0y; \\ \min_x &= \text{minimum } x \text{ value of } I_s \text{ in } x0y; \text{ and} \\ \min_y &= \text{minimum } y \text{ value of } I_s \text{ in } x0y. \end{aligned}$$

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 a_1 or to the *cardus*. Labels \blacktriangle and \blacktriangledown 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 a_4 axis labelled with \blacktriangle , erases the label on the original a_4 axis and creates a new \blacktriangle 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 by x_0y . 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 a higher level of hierarchy axis. Rule 4c exists for this purpose and rule 4d is symmetrical to it. Steps 2-11 in the derivation 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 \blacktriangle , \blacktriangledown , \blacktriangleright and \blacktriangleleft 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, Duarte, Montenegro, Gil 2009], in which the goal of the grammar was the replication of Praia plan.

Let us again assume 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 (see fig. 5).

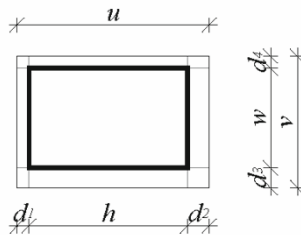
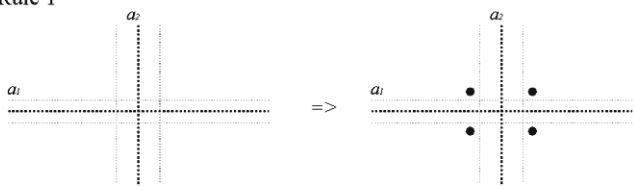
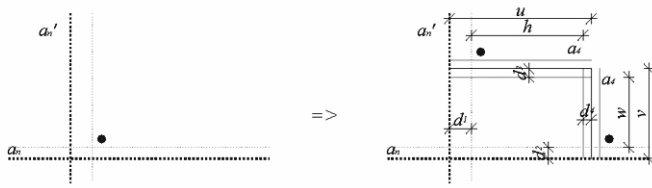
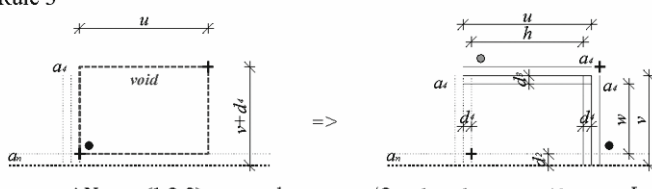
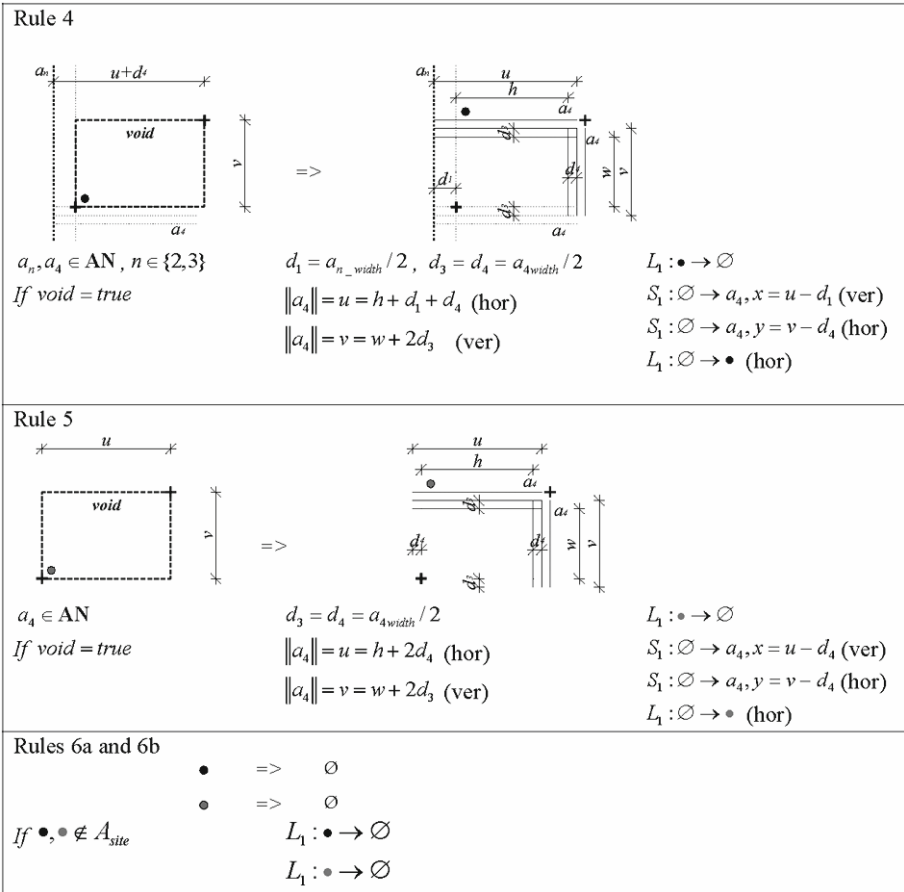


Fig. 5. The block cell – parameters and labels

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.

Fig. 6 (this page and facing page). UIP AddingBlockCells – main rules

<p>UIP – <i>AddingBlockCells</i> – main rules Parameters h and w are block length and block width respectively. The coordinates x and y refer to the left bottom corner label + of the <i>void</i> area or to two main street crossings. <i>AddingBlockCells</i>: $\Gamma^* = \{\Gamma_1\}$ $\{S_1, L_1\} \subset \Gamma_1, S_1, L_1 \in \text{AN}$</p>		
<p>Rule 1</p>  <p>$\exists a_1, a_2 \in \text{AN}$ a_1 and a_2 can also be a <i>cardus</i> and a <i>decumanus</i></p> <p style="text-align: right;">$L_1 : \emptyset \rightarrow \bullet$ (4x)</p>		
<p>Rule 2</p>  <p>$a_n, a_n', a_4 \in \text{AN},$ $n \in \{1, 2, 3\}$</p> <p>$d_1 = a_{n_width} / 2, d_2 = a_{n_width} / 2,$ $d_3 = d_4 = a_{4width} / 2$ $\ a_4\ = u = d_1 + h + d_4$ (hor) $\ a_4\ = v = d_2 + w + d_3$ (ver)</p> <p style="text-align: right;">$L_1 : \bullet \rightarrow \emptyset$ $S_1 : \emptyset \rightarrow a_4, x = u$ (ver) $S_1 : \emptyset \rightarrow a_4, y = v$ (hor) $L_1 : \emptyset \rightarrow \bullet$ (ver) $L_1 : \emptyset \rightarrow \bullet$ (hor)</p>		
<p>Rule 3</p>  <p>$a_n, a_4 \in \text{AN}, n \in \{1, 2, 3\}$ If <i>void</i> = true</p> <p>$d_2 = a_{n_width} / 2, d_3 = d_4 = a_{4width} / 2$ $\ a_4\ = u = h + 2d_4$ (hor) $\ a_4\ = v = d_2 + w + d_3$ (ver)</p> <p style="text-align: right;">$L_1 : \bullet \rightarrow \emptyset$ $S_1 : \emptyset \rightarrow a_4, x = u - d_4$ (ver) $S_1 : \emptyset \rightarrow a_4, y = v - d_2$ (hor) $L_1 : \emptyset \rightarrow \bullet$ (ver) $L_1 : \emptyset \rightarrow \bullet$ (hor)</p> <p>Note: The + mark in the <i>void</i> vertices is used just to increase legibility and should not be read as part of the rules.</p>		



Liew points out three main problems in the use of shape grammars: “(1) controlling rule selection and sequencing in a grammar; (2) filtering out information in a drawing for rule application; and (3) specifying contextual requirements of a schema” [Liew 2004: 14]. He provides seven descriptors to be used in the rule application process to solve some of these problems. 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 or specific shapes for the rule to apply. Because of the unpredictability of the *AddingBlockCells* grammar a similar descriptor needs to be 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 as in [Beirão, Duarte, Montenegro, Gil 2009] but the descriptor zone was added to three of them (see fig. 6).

Rule 1 (fig. 6) places 4 labels \bullet in the intersection of an a_1 and an a_2 axis, or in the intersection of a *cardus* and a *decumanus*. Rule 2 starts the cell derivation, erasing one of the labels \bullet 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 the *void* predicate is no longer satisfied. The values for w and h are randomly chosen from the admissible range

in each iteration. To ensure the 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 left top corner of the cell above the new a_4 axis, to be used later by Rule 5. Labels ● are only recognized by Rule 5 and their adjustment variations (as shown in fig. 7). To summarise, Rule 3 creates cells along the a_j 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. 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 (fig. 7). 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 a_n 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 here as there are no existing buildings in the Praia site.

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 on the top of a_n' to allow another generation sequence to start in another area of the plan (see derivation in fig. 8). 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. They are:

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.

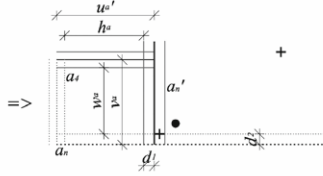
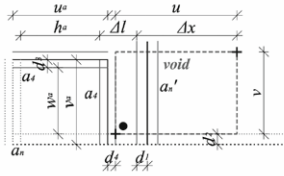
Rule 5A_8 detects the presence of two a_4 axes and one a_n axis.

Rule 5A_1 detects the Δx length of penetration of 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.

Fig. 7 (this page and following 4 pages). UIP AddingBlockCells (continuation) – adjustment rules.
 When the void predicate is not satisfied, the main rules adjust to the conditions of the context

UIP – <i>AddingBlockCells</i> (continuation) – adjustment rules		
Parameters h and w are block length and block width respectively. The coordinates x and y refer to the left bottom corner label $+$ of the <i>void</i> area or to two main street crossings. In order to simplify the notation, label coordinates were omitted from the rules.		
Rule 3A_1		
<p><i>If</i></p> <p>$void = false \wedge \Delta x \leq u/2$</p> <p>$a_n, a_n', a_4 \in \mathbf{AN}$,</p> <p>$n \in \{1, 2, 3\}$</p>	<p>$d_1 = a_{n'} _width / 2, \quad d_2 = a_{n_width} / 2,$</p> <p>$d_3 = d_4 = a_{4_width} / 2$</p> <p>$\ a_4\ = u' = d_1 + h' + d_4 \quad v = d_2 + w + d_3$</p> <p>$h' = u - \Delta x \quad u = h + 2d_4$</p>	<p>$L_1 : \bullet \rightarrow \emptyset$</p> <p>$S_1 : \emptyset \rightarrow a_4, y = v - d_2$ (hor)</p> <p>$L_1 : \emptyset \rightarrow \bullet$ (ver)</p> <p>$L_1 : \emptyset \rightarrow \circ$ (hor)</p>
Rule 3A_2		
<p><i>If</i></p> <p>$void = false \wedge \Delta x > u/2$</p> <p>$a_n, a_n', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$</p>	<p>$d_1 = a_{n'} _width / 2, \quad d_2 = a_{n_width} / 2,$</p> <p>$d_3 = d_4 = a_{4_width} / 2$</p> <p>$u = h + 2d_4 \quad v = w + 2d_4$</p> <p>$u_a = h_a + 2d_4 \quad v_a = w_a + d_3 + d_2$</p> <p>$\ a_4\ = u_a' = u_a + (u - \Delta x) + d_1 + d_4$</p> <p>$h_a' = u_a + (u - \Delta x) \quad u \neq u_a, v \neq v_a$</p>	<p>$L_1 : \bullet \rightarrow \emptyset$</p> <p>$S_1 : a_4 \rightarrow a_4, u_a \rightarrow u_a'$ (hor)</p> <p>$S_1 : a_4 \rightarrow \emptyset$ (ver)</p> <p>$L_1 : \emptyset \rightarrow \bullet$ (ver)</p>
Rule 3A_3		
<p><i>If</i></p> <p>$void = false \wedge \Delta x \leq u/2 \wedge a_n = a_1$</p> <p>$a_n, a_n', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$</p>	<p>$d_1 = a_{n'} _width / 2, \quad d_2 = a_{n_width} / 2,$</p> <p>$d_3 = d_4 = a_{4_width} / 2$</p> <p>$u = h + 2d_4 \quad v = w + d_3 + d_2$</p> <p>$\ a_4\ = u' = u - d_4 + d_1$ (hor)</p> <p>a_n' moves $\Delta x - 2d_4$ in positive direction</p>	<p>$L_1 : \bullet \rightarrow \emptyset$</p> <p>$S_1 : \emptyset \rightarrow a_4, y = v - d_2$ (hor)</p> <p>$S_1 : a_n' \rightarrow a_n'$</p> <p>$x \rightarrow x + \Delta x - 2d_4$</p> <p>$L_1 : \emptyset \rightarrow \bullet$ (ver)</p> <p>$L_1 : \emptyset \rightarrow \circ$ (hor)</p>

Rule 3A_4

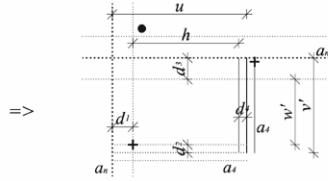
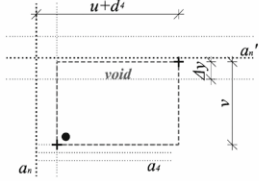


If $void = false \wedge \Delta x > u/2 \wedge a_n = a_1$
 $a_n, a_n', a_4 \in \mathbf{AN}$

$$\begin{aligned} d_1 &= a_n'_{width}/2, \quad d_2 = a_n_{width}/2, \\ d_3 &= d_4 = a_{4width}/2 \\ u &= h + 2d_4 \quad v = w + 2d_4 \\ u_a &= h_a + 2d_4 \quad v_a = w_a + d_2 + d_3 \\ \|a_4\| &= u_a' = u_a - d_4 + d_1 \quad (\text{hor}) \\ \Delta l &= 2d_4 + (u - \Delta x) \quad u \neq u_a, v \neq v_a \\ a_n' &\text{ moves } \Delta l \text{ in negative direction} \end{aligned}$$

$L_1 : \bullet \rightarrow \emptyset$
 $S_1 : a_4 \rightarrow \emptyset$ (ver)
 $S_1 : a_n' \rightarrow a_n'$
 $x \rightarrow x - \Delta l$
 $S_1 : a_4 \rightarrow a_4, u_a \rightarrow u_a'$ (hor)
 $L_1 : \emptyset \rightarrow \bullet$ (ver)

Rule 4A_1

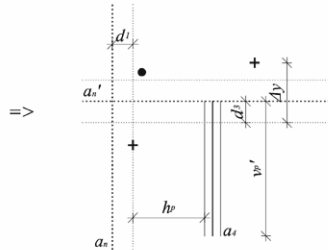
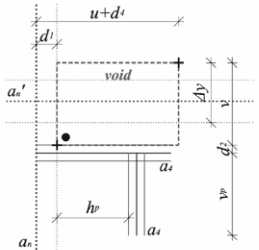


If $void = false \wedge \Delta y \leq v/2$
 $a_n, a_n', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$

$$\begin{aligned} d_3 &= a_n'_{width}/2, \quad d_4 = a_n_{width}/2, \\ d_2 &= d_4 = a_{4width}/2 \\ u &= h + d_4 + d_1 \quad v = w + 2d_4 \\ \|a_4\| &= v' = v + d_4 + d_3 - \Delta y \\ w' &= w - \Delta y + d_4 \end{aligned}$$

$L_1 : \bullet \rightarrow \emptyset$
 $S_1 : \emptyset \rightarrow a_4, x = u - d_1$ (ver)
 $L_1 : \emptyset \rightarrow \bullet$

Rule 4A_2

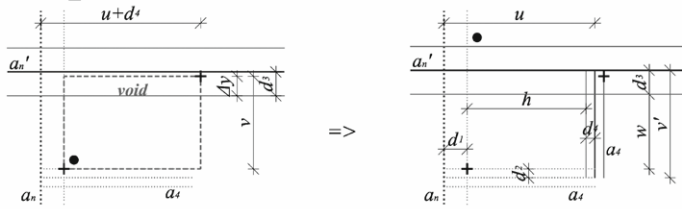


If $void = false$
 $\wedge \Delta y > v/2$
 $a_n, a_n', a_4 \in \mathbf{AN},$
 $n \in \{1, 2, 3\}$

$$\begin{aligned} d_3 &= a_n'_{width}/2, \quad d_1 = a_n_{width}/2, \quad d_2 = d_4 = a_{4width}/2 \\ u &= h + d_4 + d_1 \quad v = w + 2d_4 \quad h \neq h_p \\ \|a_4\| &= v_p' = v_p + (v - \Delta y) + d_2 + d_3 \quad (\text{ver}) \\ w_p' &= w_p + (v - \Delta y) + d_2 + d_3 \\ a_n' &\text{ moves } \Delta x - 2d_4 \text{ in positive direction} \end{aligned}$$

$L_1 : \bullet \rightarrow \emptyset$
 $S_1 : a_4 \rightarrow \emptyset$ (hor)
 $S_1 : a_4 \rightarrow a_4, v_p \rightarrow v_p'$ (ver)
 $L_1 : \emptyset \rightarrow \bullet$

Rule 4A_3



If
 $void = false \wedge \Delta y \leq v/2$
 $\wedge a_n = decumanus$
 $a_n, a_n', a_4 \in \mathbf{AN}$,
 $n' \in \{2,3\}$

$$d_3 = a_{n'.width}/2, \quad d_1 = a_{n'.width}/2,$$

$$d_2 = d_4 = a_{4.width}/2$$

$$u = h + d_4 + d_1 \quad v = w + 2d_2$$

$$\|a_4\| = v' = v - \Delta y + d_2 + d_3 \quad (\text{ver})$$

a_n' moves $\Delta y - 2d_2$ in positive direction

$$L_1 : \bullet \rightarrow \emptyset$$

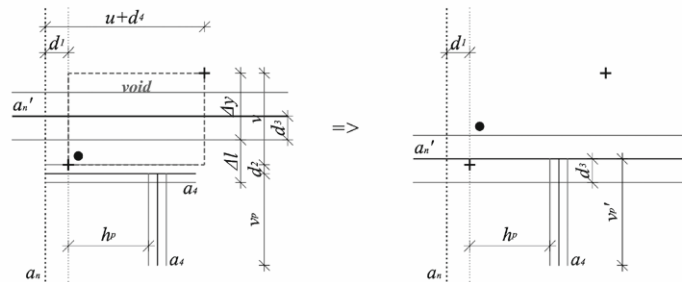
$$S_1 : a_n' \rightarrow a_n'$$

$$y \rightarrow y + \Delta y - 2d_2$$

$$S_1 : a_4 \rightarrow a_4, x = u - d_1 \quad (\text{ver})$$

$$L_1 : \emptyset \rightarrow \bullet$$

Rule 4A_4



If
 $void = false$
 $\wedge \Delta y > v/2$
 $\wedge a_n = decumanus$
 $a_n, a_n', a_4 \in \mathbf{AN}$,
 $n' \in \{2,3\}$

$$d_3 = a_{n'.width}/2, \quad d_1 = a_{n'.width}/2,$$

$$d_2 = d_4 = a_{4.width}/2$$

$$u = h + d_4 + d_1 \quad v = w + 2d_2 \quad h \neq h_p$$

$$\|a_4\| = v_p' = v_p + d_2 + (v - \Delta y) - \Delta l + d_3 \quad (\text{ver})$$

$$\Delta l = v - \Delta y + 2d_2$$

a_n' moves Δl in negative direction
 h_p and w_p are the block length and width
of the previous cell

$$L_1 : \bullet \rightarrow \emptyset$$

$$S_1 : a_4 \rightarrow \emptyset \quad (\text{hor})$$

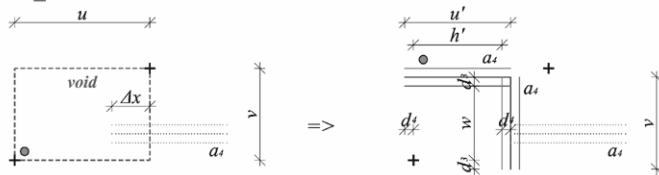
$$S_1 : a_n' \rightarrow a_n'$$

$$y \rightarrow y - \Delta y + 2d_2$$

$$S_1 : a_4 \rightarrow a_4, v_p \rightarrow v_p' \quad (\text{ver})$$

$$L_1 : \emptyset \rightarrow \bullet$$

Rule 5A_1a



If
 $void = false \wedge$
 $\Delta x \leq u/2$
 $a_4 \in \mathbf{AN}$

$$d_3 = d_4 = a_{4.width}/2$$

$$u = h + 2d_4$$

$$\|a_4\| = v = w + 2d_3 \quad (\text{ver})$$

$$\|a_4\| = u' = u - \Delta x + d_4 \quad (\text{hor})$$

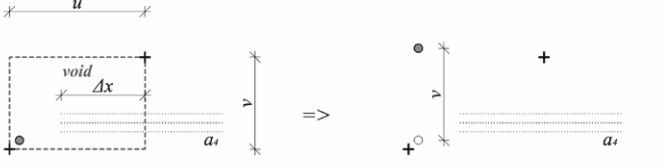
$$L_1 : \bullet \rightarrow \emptyset$$

$$S_1 : \emptyset \rightarrow a_4, y = v - d_3 \quad (\text{hor})$$

$$S_1 : \emptyset \rightarrow a_4, x = u' - d_4 \quad (\text{ver})$$

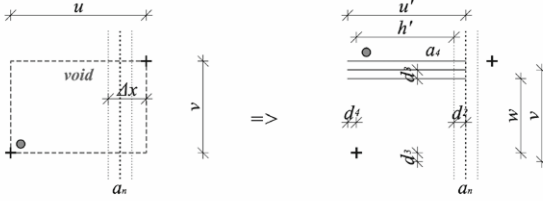
$$L_1 : \emptyset \rightarrow \bullet \quad (\text{hor})$$

Rule 5A_1b



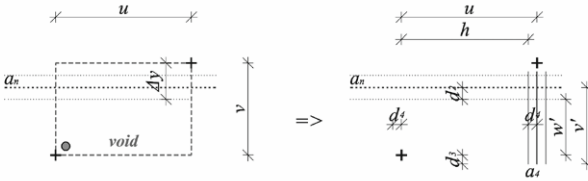
If $void = false \wedge \Delta x > u/2$ $d_3 = d_4 = a_{4width}/2$ $L_1 : \bullet \rightarrow \emptyset$
 $a_4 \in \mathbf{AN}$ $u = h + 2d_4$ $v = w + 2d_3$ $L_1 : \emptyset \rightarrow \bullet$

Rule 5A_2



If $void = false \wedge \exists a_n \in void$ $d_3 = d_4 = a_{4width}/2$ $L_1 : \bullet \rightarrow \emptyset$
 $\forall \Delta x$ $u = h + 2d_4$ $v = w + 2d_3$ $S_1 : \emptyset \rightarrow a_4, y = v - d_3$ (hor)
 $a_n \in \mathbf{AN}, n \in \{1, 2, 3\}$ $\|a_4\| = u' = u + \Delta x + d_2 + d_4$ $L_1 : \emptyset \rightarrow \bullet$ (hor)

Rule 5A_3



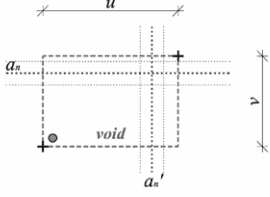
If $void = false \wedge \exists a_n \in void$ $d_3 = d_4 = a_{4width}/2$ $L_1 : \bullet \rightarrow \emptyset$
 $\forall \Delta y$ $u = h + 2d_4$ $v = w + 2d_3$ $S_1 : \emptyset \rightarrow a_4, x = u - d_4$ (ver)
 $a_n \in \mathbf{AN}, n \in \{1, 2, 3\}$ $\|a_4\| = v' = v + \Delta y + d_2 + d_3$

Rule 5A_4

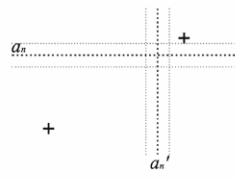


If $void = false$ $d_3 = d_4 = a_{4width}/2$ $L_1 : \bullet \rightarrow \emptyset$
 $\forall \Delta x$ $u = h + 2d_4$ $v = w + 2d_3$ $S_1 : \emptyset \rightarrow a_4, y = v - d_3$ (hor)
 $a_4 \in \mathbf{AN}$ $\|a_4\| = u' = u - \Delta x$ $h' = h - \Delta x$ $S_1 : \emptyset \rightarrow a_4, x = u - d_4$ (ver)
 $L_1 : \emptyset \rightarrow \bullet$ (hor)

Rule 5A_5



=>



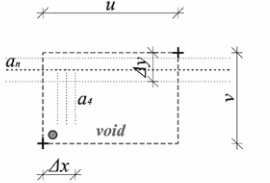
If $void = false \wedge \exists a_n, a_n' \in void$

$$d_3 = d_4 = a_{4width} / 2$$

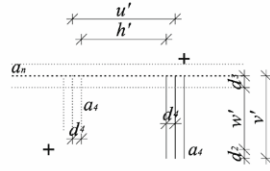
$$u = h + 2d_4 \quad v = w + 2d_3$$

$L_1 : \bullet \rightarrow \emptyset$

Rule 5A_6



=>



If $void = false \wedge \exists a_n, a_4 \in void$

$$d_3 = d_4 = a_{4width} / 2$$

$$u = h + 2d_4 \quad v = w + 2d_3$$

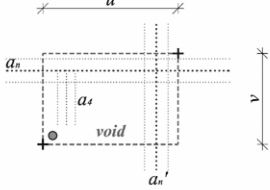
$$u' = u - \Delta x \quad h' = h - \Delta x$$

$$\|a_4\| = v' = v - \Delta y + d_2 + d_3$$

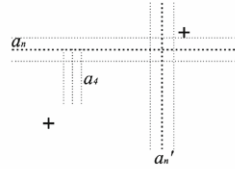
$L_1 : \bullet \rightarrow \emptyset$

$S_1 : \emptyset \rightarrow a_4, x = u - d_4$ (ver)

Rule 5A_7



=>



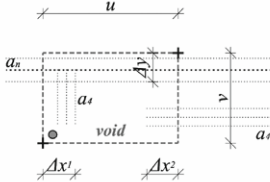
If $void = false \wedge \exists a_n, a_n', a_4 \in void$

$$d_3 = d_4 = a_{4width} / 2$$

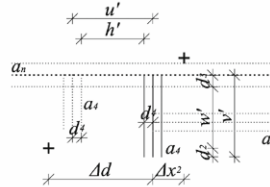
$$u = h + 2d_4 \quad v = w + 2d_3$$

$L_1 : \bullet \rightarrow \emptyset$

Rule 5A_8



=>



If $void = false$
 $\wedge \exists a_n, a_4 \in void$

$$d_2 = d_4 = a_{4width} / 2$$

$$u = h + 2d_4$$

$$u' = u - \Delta x_1 - \Delta x_2 + d_4$$

$$\|a_4\| = v' = v - \Delta y + d_2 + d_3$$

$$u = \Delta d + \Delta x_2$$

$$d_3 = a_{n_width} / 2$$

$$v = w + 2d_2$$

$$h' = h - \Delta x_1 - \Delta x_2 + d_4$$

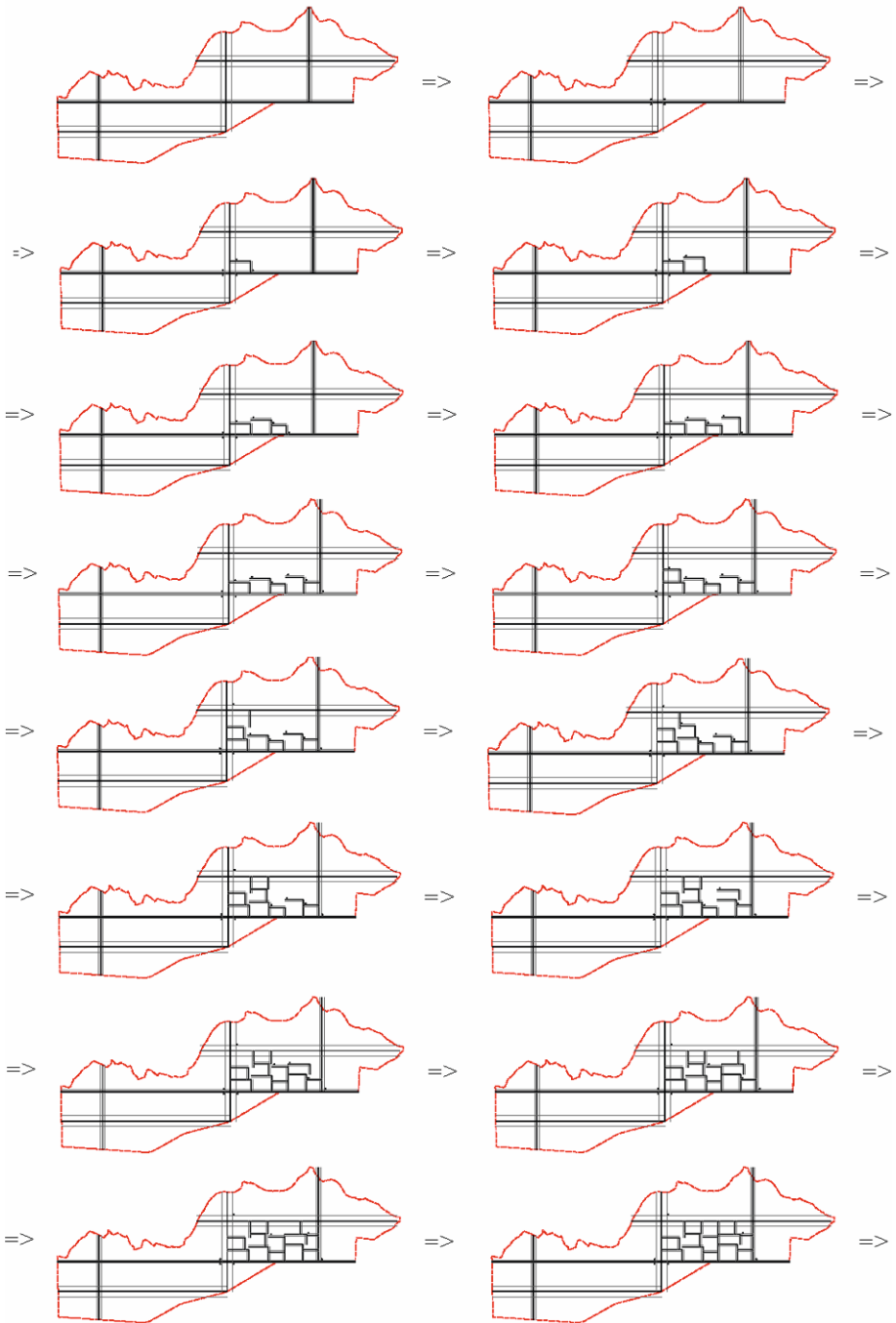
$$w' = w - \Delta y + d_2 + d_3$$

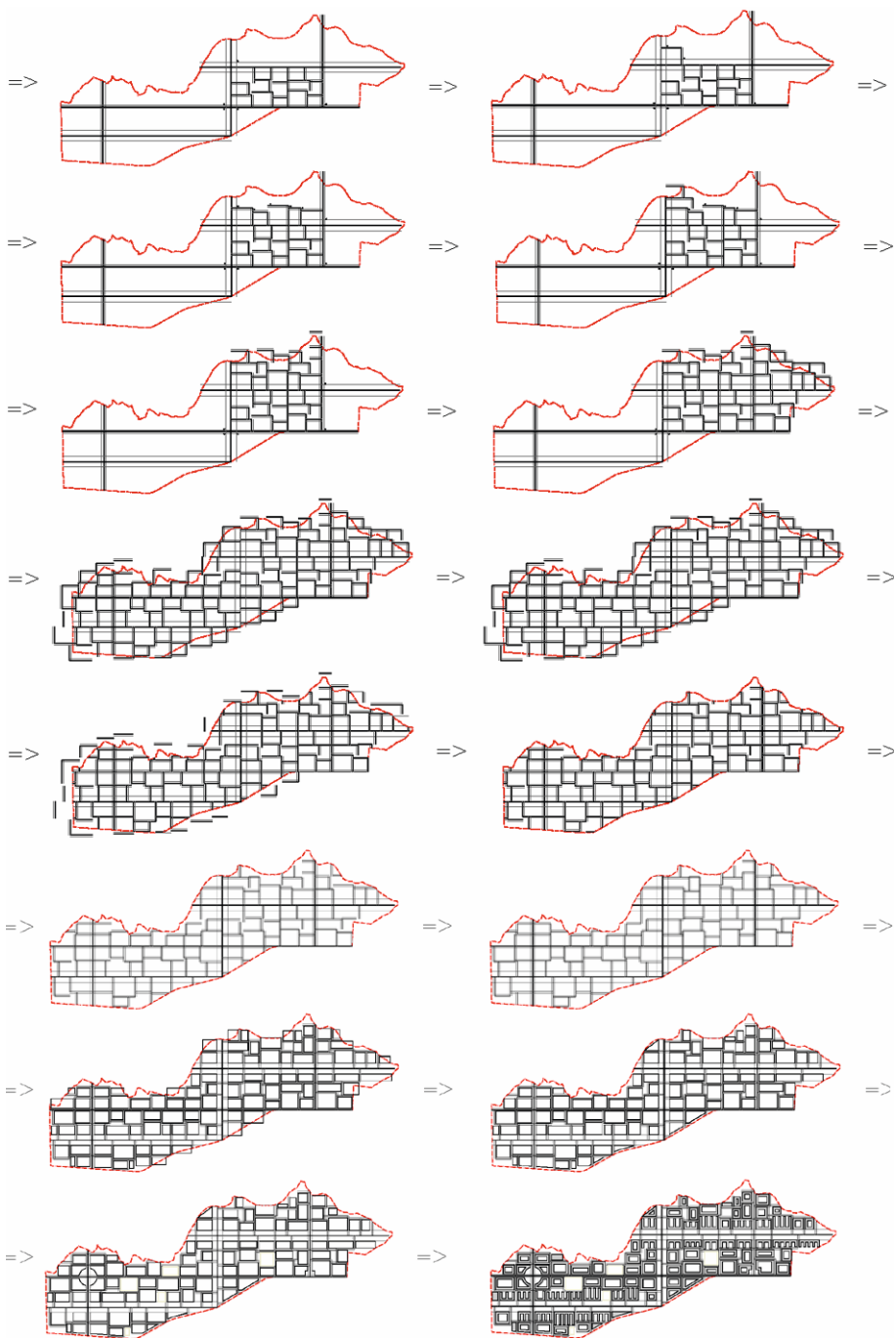
$$\Delta d = \Delta x_1 + h' + d_4$$

$L_1 : \bullet \rightarrow \emptyset$

$S_1 : \emptyset \rightarrow a_4, x = \Delta d$ (ver)

Fig. 8 (this page and facing page). AddingBlockCells – derivation including adjustment rules





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 a new label \bullet 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 \bullet and does not create a new one.

Rule 5A_4 is similar to 5A_1 but it generates the reduced cell leaning against the right side of the void *zone* instead of the left side. In fact, it is using the free space 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 \bullet . 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 recreate new \bullet labels.

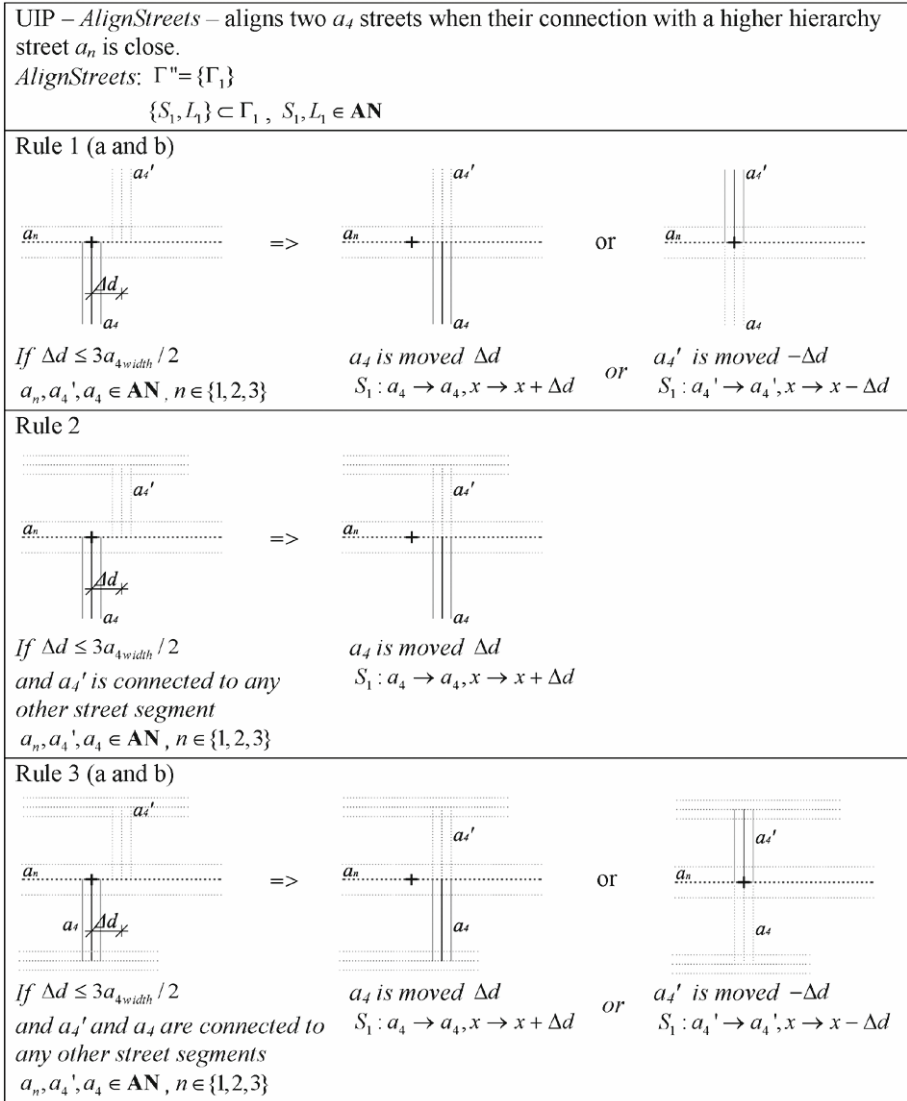
Rule 5A_9 reduces the cell length u to u' following the equation $u' = u - \Delta x_1 - \Delta x_2 + d_2 + d_4$, producing a cell with dimensions $u' \times v$.

The derivation in fig. 8 shows the generation of an urban plan using *AddingBlockCells* as the main 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 \bullet labels. Step 3 applies Rule 2, generating the first cell, erasing one of the \bullet 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 \bullet 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 \bullet 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 \bullet labels, starting with Rule 2 and ending with the exhaustion of labels \bullet and \circ . Note that every label falling outside I_s is erased. Step 23 is the last cell creation step.

The whole set of rules allows the urban grid to be generated without conflicts but the final results still need adjustments, namely aligning or connecting a few streets and placing blocks within cells. However, these adjustments are produced by other UIPs. The derivation in fig. 8 clearly shows some of these problems in steps 23 to 26.

In step 24 the UIP *AlignStreets* is applied. The result of the generation using *AddingBlockCells* can produce several situations where a_4 axes connect to a_n axes that are very close to each other but not actually aligned. *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 crossroads 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 and degrees of connectivity used in this UIP are as follows, increasing from (1) to (3): (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 fig. 9).

Fig. 9 (this page and following 2 pages). UIP AlignStreets – Aligns two a_4 streets when connecting to a higher hierarchy street. Moves the street with less connections or alignments in the street network



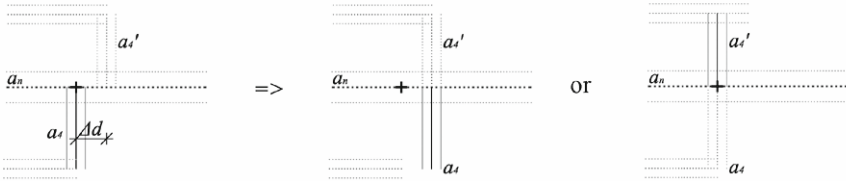
Rule 4



If $\Delta d \leq 3a_{4width}/2$
 and a_4' has a corner
 connection to
 any other street segment
 $a_n, a_4', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$

a_4 is moved Δd
 $S_1: a_4 \rightarrow a_4, x \rightarrow x + \Delta d$

Rule 5 (a and b)

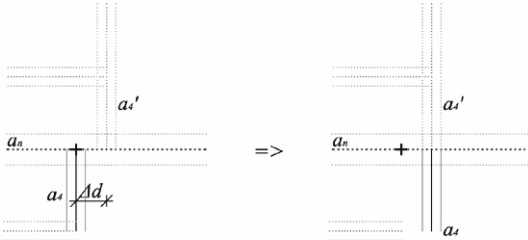


If $\Delta d \leq 3a_{4width}/2$
 and a_4' and a_4 have a corner
 connection to other street
 segments
 $a_n, a_4', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$

a_4 is moved Δd
 $S_1: a_4 \rightarrow a_4, x \rightarrow x + \Delta d$

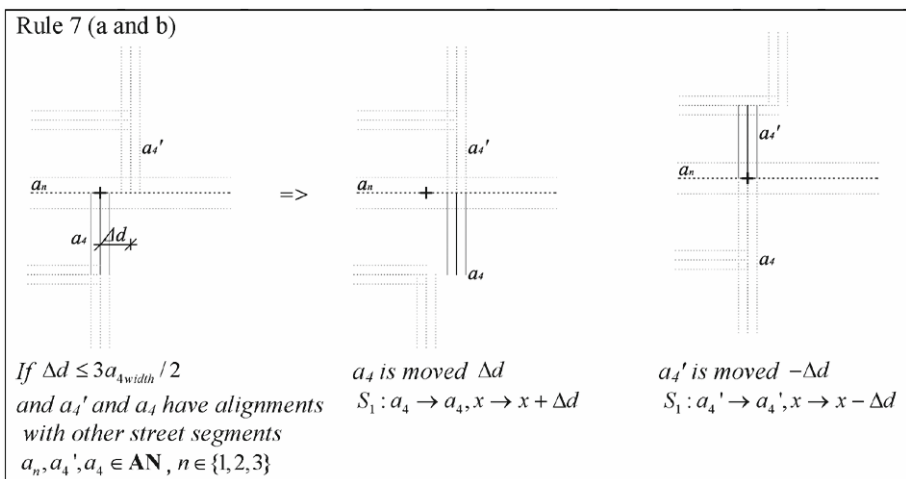
OR
 a_4' is moved $-\Delta d$
 $S_1: a_4' \rightarrow a_4', x \rightarrow x - \Delta d$

Rule 6



If $\Delta d \leq 3a_{4width}/2$
 and a_4' has an alignment with
 any other street segment
 $a_n, a_4', a_4 \in \mathbf{AN}, n \in \{1, 2, 3\}$

a_4 is moved Δd
 $S_1: a_4 \rightarrow a_4, x \rightarrow x + \Delta d$

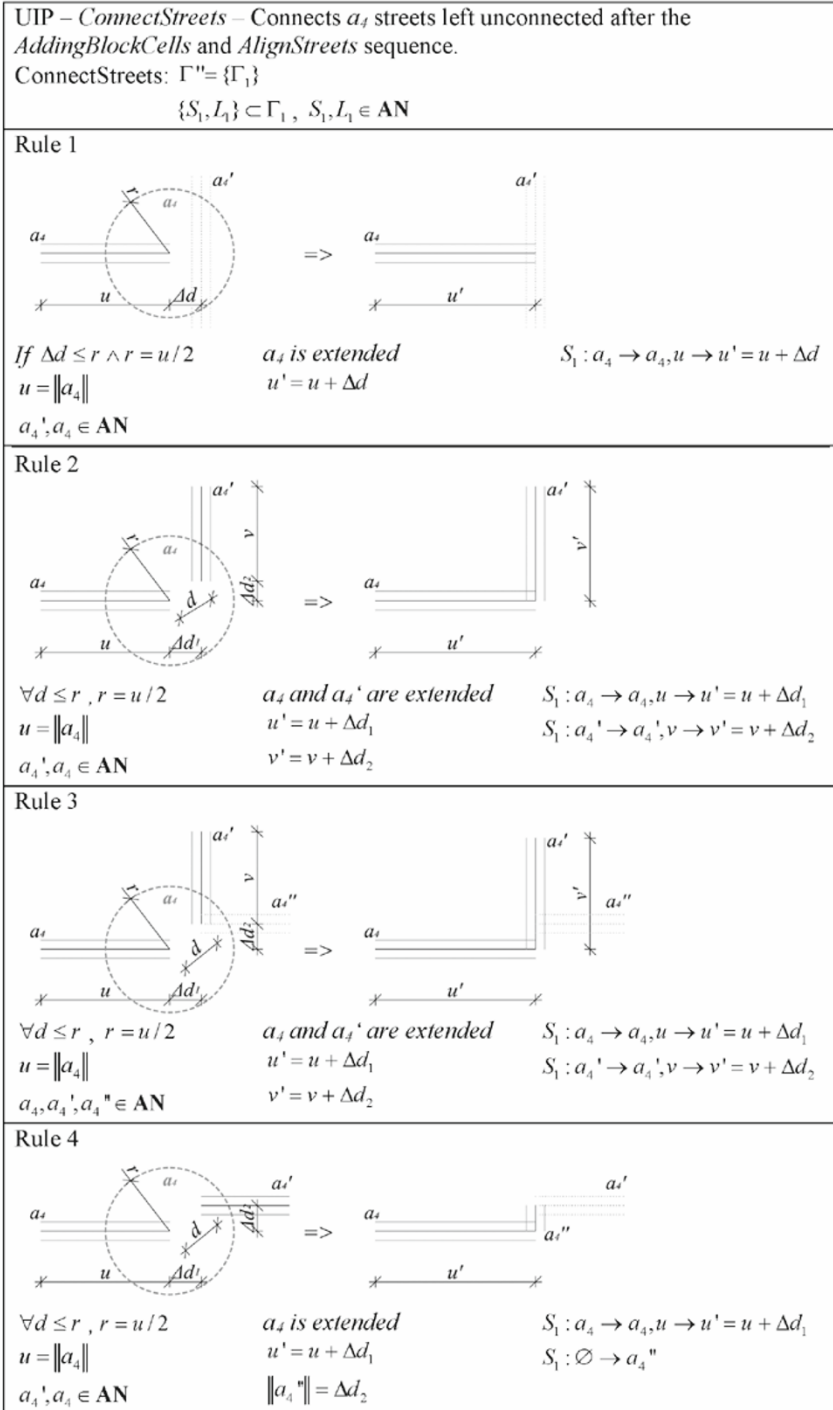


In steps 25-26 all the axes falling outside I_s are either trimmed (step 25) or erased (step 26). Very small bits of a_4 axes are left from step 25. All the bits smaller than half of the lower value defined for h are erased (step 27). In this step it can be seen that some of the streets are still not connected and do not contribute towards the consistency of the street network. *ConnectStreets* is used to correct some of these inconsistencies in the grid generation (fig. 10). Step 28 shows the result of applying this UIP. In step 29 the cells are filled with abstract blocks using *AddBlocktoCells* [Beirão, Duarte, Montenegro, Gil 2009]. Step 30 adjusts or erases the blocks on the borders of the plan. Every block falling outside the I_s area is erased. In step 31 we can see the final result of applying several UIPs to square or public space generation. Steps 32-33 replace the abstract block with a few different block types showing the different possibilities for building occupation within the block. The last steps and UIPs are not detailed or discussed in this paper. However, 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 given generation, but rather a demonstration of the versatility of the UIP *AddingBlockCells* in the generation of plans. This is accomplished simply 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 the formulation module, in order to generate solutions following specific criteria.

Finally, we need to point 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. This structure is prepared for direct export to a GIS platform so that an integrated analysis can be performed within the context of the plan. 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.

Fig. 10. UIP ConnectingStreets – Connects a_4 streets left unconnected after the *AddingBlockCells* and *AlignStreets* sequence



6 Discussion

The purpose of this paper is to show the versatility of design defined within the scope of the City Induction generation module by using an approach based on shape grammar. This is obtained at the level of grid generation simply by manipulating the range of admissible values for two parameters: the block length h and the block width w . The plan generation mechanisms are shape grammars encoding typical urban design moves. These grammars were called Urban Induction Patterns and designs are generated through the application of different arrangements of these UIPs. In this paper we show that the same rules used to generate one of the case studies – Plan 1 – can actually be used to generate other results, depending on how the available options in the system are manipulated, namely (1) to define a specific (sequential) arrangement of UIPs and (2) to manipulate the available rule parameters.

Current research is engaged in showing that a rather small set of UIPs can be used to generate a large amount of variety in design, including the scope of the four case studies. This paper already envisages that by using specific values for the parameters h and w for each iteration of *AddingAxes*, the grid of the Ijburg plan can be obtained (see plan 3 in fig. 2). In addition, a close look at Plan 2 shows that it uses approximately the same UIPs as Plan 1. This should not seem strange considering that the plans come from the same author but, nevertheless, careful examination shows that Plan 2 is divided into four sectors and each sector uses similar rule sequences. Only the parameters have considerably more variation. In each sector one main axis can clearly be seen with at least one orthogonal axis, a promenade and an orthogonal grid in which even some of the block types are similar. Hardly any new UIPs or new rules need to be added.

The interest of *AddingBlockCells* seems less evident in terms of grid generation, particularly since the example shown in fig. 8 looks a little unusual and maybe even a little messy. However, if we consider that instead of applying this UIP to highly structured top-down approaches such as the classic *cardus* and *decumanus* urban plans, the same or similar rules could be used to apply a bottom-up approach, i.e., generation starting from some focal point in an open area that progressively grows by adding cells step by step. This approach would definitely increase the interest of such an UIP, particularly if combined with different definitions for the urban units occupying the cells. Although some work is already being developed in this area, it certainly appears to be a promising research field in which there is still a lot to explore. Another important aspect is that it is evident that each grid generator UIP has a character of its own. *AddingAxes* generates a typical iron grid plan while *AddingBlockCells* generates a more informal urban fabric leaving some small public spaces with spatial characteristics that resemble certain areas in traditional cities. Furthermore, it can even be envisaged that both UIPs could be applied to different sectors of the plan, enabling a plan to be composed by developing areas with different characters.

The exploration of rules for block types is also clearly an interesting domain for further research. The last steps in the derivations shown here were not detailed but the rules are, in fact, very simple and can be roughly described as variations on the simplified rules shown in Figure 6 in Beirão et al [2008]. Pedro [1999] also suggests that most block occupations are variations or compositions based on only 3 types of block occupation: a closed block with a peripheral occupation, a linear block composed of parallel building blocks or an individual block composed of isolated buildings. The combination of these 3 types may show that most block compositions are actually based

on very simple rules and the composition of very few basic elements. However, the Dutch case studies clearly show that block types can be very complex urban units and may give rise to the development of a few more UIPs to generate more complexity in the design at local level.

The main interest of this discussion is to support the hypothesis that a generation system based on the principles described in this paper can in fact be much simpler than initial examination may suggest. Future research will aim to prove this hypothesis, i.e., that a small set of UIPs with some available open parameters may be sufficient to provide a highly versatile design tool with generative capacities that can be easily integrated into GIS platforms. Furthermore, considering that design languages are personal expressions of designers, we would like to extend the goals of City Induction to provide ways of easily defining new UIPs or customizing existing ones in order to make not only an extremely generic, but also a highly customizable urban grammar available.

One of the most promising aspects of the use of the ontology which needs further exploration is the use of the relationships between the spatial components of the ontology through clear definitions of inheritance relations between object classes, relating these inheritance relationships to the UIPs in order to provide an automated means of reducing the set of available UIPs to be applied during the consecutive steps of the derivation without the need to establish a specific algorithm for this purpose. This is also related to the concept of a methodological approach to urban design which is part of the task of this research to clarify. A future paper will address this subject.

3D representations have not been mentioned so far, but have not been forgotten. There are two different aspects to this subject. One simply concerns giving volume to the construction, which may easily be achieved by simply extruding the building polygons. This later needs to be linked to the generation process as the construction volume is directly related to urban parameters such as density, which is likely to be taken as a generation goal. The second aspect concerns topography and in this regard several cumulative issues might need to be taken into account. Two main categories of problem definitely emerge: (1) a methodological one, which may define a procedure to select the areas to be planned and urbanized by restricting the intervention area to a smaller zone or zones subject to distinct rules depending on slope, and (2) a rule-based approach to deal with topography and the design of streets and buildings in steep areas.

Finally, we need to remember that this paper focuses exclusively on the structure of the generation module. We should not forget that this is simply the design mechanism of a larger tool aimed at integrating programme formulation and evaluation techniques within the generation of designs. Additional functions emerging from the other two modules will improve the accuracy and credibility of the designs generated, as they may guide generation towards better solutions. It is planned to integrate the three modules supported by the said spatial ontology. However, this ontology still needs further detailing in order to integrate relationships between the morphological structures and the social and economical concepts used in formulation and evaluation processes. In addition, even at the morphological level further detailing is envisaged in order to approach the lower scale detailing of the plans.

7 Conclusions

Although it may be said that a design is the result of a sequence of design moves, it is not desirable to develop systems that impose a specific sequence for the simple reason that such decisions are context dependent. Urban Grammars, being an arrangement of generic grammars, allow for greater design flexibility in the sense that the sequence of procedures is not predefined but is instead defined by the designer during the design process. This paper demonstrates how a design tool is defined using generic grammars for the creation of customized urban designs produced by a customizable design language, a specific compound grammar defined by an arrangement of generic grammars.

The generation module for City Induction provides a very generic Urban Grammar composed of several generic grammars called Urban Induction Patterns corresponding to typical urban design moves. Specific grammars, such as the Praia grammar or the Ijburg grammar can be obtained by defining specific arrangements of Urban Induction Patterns and specific constraints on the rule parameters. As demonstrated in this paper, variations on the UIP arrangements or the rule parameters can provide design variations and allow for design exploration.

Acknowledgments

The City Induction project is funded by the Fundação para a Ciência e Tecnologia (FCT), Portugal (PTDC/AUR/64384/2006), hosted by ICIST at TU Lisbon, and coordinated by José P. Duarte. J.N. Beirão is also funded by the FCT by grant SFRH/BD/39034/2007. Beirão, J. Gil, and N. Montenegro are responsible for the generation, the evaluation, and the formulation modules, respectively. We would like to thank Henco Bekkering, Sevil Sariyildiz and Frank van der Hoeven for their readings, comments and support. Thank you to George Stiny and Terry Knight.

Notes

1. Zones indicating areas with specific features are represented in shape-files, as well as representations of real objects, but they are conceptual representations of our environment.
2. 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.

References

- ALEXANDER, Christopher, Sara ISHIKAWA and Murray SILVERSTEIN. 1977. *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press.
- ASCHER, F. 2001. *Les nouveaux principes de l'urbanisme. La fin des villes n'est pas à l'ordre du jour*. La Tour d'Aigues: Éditions de l'Aube.
- BATTY, M. 2005. *Cities and Complexity: Understanding Cities with Cellular Automata, AgentBased Models, and Fractals*. Cambridge, MA: MIT Press.
- BEIRÃO, José Nuno and José Pinto DUARTE. 2009. Urban Design with Patterns and Shape Rules. Pp. 148-165 in *Model Town: Using Urban Simulation in New Town Planning*, EH Stolk and M Brömmelstroet, eds. 2nd International Seminar, 2007, Almere. Almere, The Netherlands: Martien de Vletter.
- BEIRÃO, José; José Pinto DUARTE and Rudi STOUFFS. 2008. Structuring a Generative Model for Urban Design: Linking GIS to Shape Grammars. Pp. 929-938 in *Architecture 'in Computro'* (Proceedings of the 26th eCAADe Conference, Antwerp, 17-20 September 2008), Marc Muylle, ed. Antwerp.
- . 2009a. Grammars of designs and grammars for designing. In *Joining Languages, Cultures and Visions: CAAD Futures 2009*, T. Tidafi and T. Dorta, eds. Montreal: University of Montreal.

- http://cumincad.scix.net/cgi-bin/works/Show?cf2009_890
- . 2009b. An Urban Grammar for Praia: Towards Generic Shape Grammars for Urban Design. Pp. 575-584 in *Computation: The New Realm of Architectural Design* [Proceedings of the 27th eCAADe Conference, Istanbul, 16-19 September 2009. Istanbul: eCAADe, YTU, ITU.
- BEIRÃO, José N., José Pinto DUARTE, Nuno MONTENEGRO and Jorge GIL. 2009. Monitoring urban design through generative design support tools: a generative grammar for Praia. Pp. 1223-1252 in *Proceedings of the 15th APDR Congress on Networks and Regional Development*. Cidade da Praia Cape Verde: APDR.
- BEIRÃO, José Nuno; Nuno MONTENEGRO; Jorge GIL; José P. DUARTE; Rudi STOUFFS. 2009. The city as a street system: A street description for a city ontology. Pp. 132-134 in *SIGraDi 2009 - Proceedings of the 13th Congress of the Iberoamerican Society of Digital Graphics*, Sao Paulo, Brazil, November 16-18, 2009. São Paulo: eCAADe, Universidade Presbiteriana MacKenzie.
- BROWN, F. E. and JOHNSON J. H. 1984. An interactive computer model of urban development: the rules governing the morphology of mediaeval London. *Environment and Planning B: Planning and Design* 12, 4: 377-400.
- DUARTE, José Pinto 2001. Customizing Mass Housing: a discursive grammar for Siza's Malagueira house. Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge.
- DUARTE, José Pinto, J. ROCHA, G. DUCLA-SOARES. 2007. Unveiling the structure of the Marrakech Medina: A Shape Grammar and an Interpreter for Generating Urban Form. *AI EDAM Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 21: 1-33.
- FLEISCHER, A. 1992. Grammatical architecture? *Environment and Planning B: Planning and Design* 19, 2: 221-226.
- FRIEDMAN, A. 1997. Design for Change: Flexible Planning Strategies for the 1990s and Beyond. *Journal of Urban Design* 2, 3: 277-295.
- GAMMA, Erich, Richard HELM, Ralph JOHNSON and John VLISSIDES. 1995, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.
- GIL, Jorge. and DUARTE, José Pinto. 2008. Towards an Urban Design Evaluation Framework. Pp. 257-264 in *Architecture 'in Computro'* (Proceedings of the 26th eCAADe Conference, Antwerp, 17-20 September 2008), Marc Muylle, ed. Antwerp.
- HILLIER, Bill. 1998. *Space is the Machine : A Configurational Theory of Architecture*, Cambridge University Press.
- HILLIER, Bill and Julianne HANSON. 1989. *The Social Logic of Space*. Cambridge: Cambridge University Press.
- KNIGHT, Terry Weissman. 2003. Computing with ambiguity. *Environment and Planning B: Planning and Design* 30, 2: 165-180
- LIEW, Haldane. 2004. SGML: A Meta-Language for Shape Grammars. Ph.D. Dissertation, Massachusetts Institute of Technology.
- MAYALL, Kevin and G. Brent HALL. 2005. Landscape grammar 1: spatial grammar theory and landscape planning. *Environment and Planning B: Planning and Design* 32, 6: 895-920.
- MONTENEGRO, Nuno C. and José Pinto DUARTE. 2008. Towards a Computational Description of Urban Patterns. Pp. 239-248 in *Architecture 'in Computro'* (Proceedings of the 26th eCAADe Conference, Antwerp, 17-20 September 2008), Marc Muylle, ed. Antwerp.
- PARISH, Yoav I. H. and Pascal MÜLLER. 2001, Procedural modeling of cities. Pp. 301-308 in *Proceedings of ACM SIGGRAPH 2001*, E. Fiume, ed. New York: ACM Siggraph.
- PEDRO, J. Branco. 1999. *Vizinhança Próxima* (Housing program. Neighbourhood). Lisbon: LNEC. (Collection Architecture Technical Information, n.º 7).
- PORTUGALI, Juval. 1999. *Self Organization and the City*. Heidelberg: Springer.
- SCHÖN, Donald A. 1983. *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books.
- STINY, George. 1980a. Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design* 7, 3: 343-351.
- . 1980b. Kindergarten grammars: designing with Froebel's building gifts. *Environment and Planning B: Planning and Design* 7, 4: 409-462.

- . 1981. A note on the description of designs. *Environment and Planning B: Planning and Design* 8, 3: pp. 257-267.
- . 2005. *Shape, talking about seeing and doing*, Cambridge, MA: MIT Press.
- STINY, George and James GIPS. 1972. Shape Grammars and the Generative Specification of Painting and Sculpture. *Information Processing* 71: 1460-1465.
- TEELING, Catherine. 1996. Algorithmic Design: Generating Urban Form. *Urban Design Studies* 2: 89-100.
- YUE, K., R. KRISHNAMURTI and F. GROBLER. 2009. Computation-friendly shape grammars: Detailed by a sub-framework over parametric 2D rectangular shapes. Pp. 757-770 in *Joining Languages, Cultures and Visions: CAAD Futures 2009*, T. Tidafi and T. Dorta, eds. Montreal: Les Presses de l'Université de Montreal.

About the authors

José Nuno Beirão was awarded a professional degree in architecture from the Faculty of Architecture of the Technical University of Lisbon in 1989. He has practiced architecture and urban design since then. After working at the Falcão de Campos and Gonçalo Byrne Offices, he started the architectural firm B Quadrado Arquitectos with Miguel S. Braz in 1998. Their work and portfolio is available at www.bquadrado.com. José Nuno Beirão is now working towards his PhD dissertation at the TU Delft Faculty of Architecture. The research follows the subject of his Master's thesis ("Urban Grammars: Towards Flexible Urban Design," ISCTE - Instituto Superior das Ciências do Trabalho e da Empresa, Lisbon, 2005) and is part of a larger research project called City Induction hosted at the TU Lisbon. His research interests are the development of customizable and flexible design systems, and have focused on housing since 1998 and more intensively on urban design since 2001. His current interests are focused on the development of shape grammars for urban design and on the use of the generative capabilities of shape grammars to support the urban design process and foster design exploration.

José Pinto Duarte holds a B.Arch. (1987) in architecture from the Technical University of Lisbon and an S.M.Arch.S. (1993) and a Ph.D. (2001) in Design and Computation from MIT. He is currently Visiting Scientist at MIT, Associate Professor at the Technical University of Lisbon Faculty of Architecture, and a researcher at the Instituto Superior Técnico, where he founded the ISTAR Labs - IST Architecture Research Laboratories. He is the co-author of *Collaborative Design and Learning* (with J. Bento, M. Heitor and W. J. Mitchell, Praeger 2004), and *Personalizar a Habitação em Série: Uma Gramática Discursiva para as Casas da Malagueira* (Fundação Calouste Gulbenkian, 2007). He was awarded the Santander/TU Lisbon Prize for Outstanding Research in Architecture by the Technical University of Lisbon in 2008. His main research interests are mass customization with a special focus on housing, and the application of new technologies to architecture and urban design in general.

Rudi Stouffs is Associate Professor of Design Informatics and leader of the Computation & Performance research group and programme at the Faculty of Architecture, Delft University of Technology. He holds an MSc in architectural engineering from the Vrije Universiteit Brussel, an MSc in computational design and a Ph.D. in architecture from the Carnegie Mellon University (CMU). He has been an Assistant Professor in the Department of Architecture at the CMU and Research Coordinator of the Architecture and CAAD course at ETH Zurich. His research interests include computational issues in description, modelling, and representation for design in the areas of information exchange, collaboration, shape recognition and generation, geometric modelling, and visualization.