primitive design elements. But what makes design in any field such an outstanding and surprising activity is the fact that rules are constantly being recreated by new authors to develop their own personal styles or design expressions..

Considering the above, two main issues are of interest for us:

- It seems by historical evidence that successful designers define their own sets of rules and primitive composition elements and that their design styles evolve by transforming and exploring transformations within their own design domain (Knight, 1981). Many examples may be found from classical times to modern times, e.g., Vitruvius, Palladio, F. L. Wright, Le Corbusier, just to list some well-known cases. (See also Knight, 1983, regarding the case of F. L. Wright's transition from the Prairie houses to the Usonian houses).
- Considering just the field of urban design, it seems possible to find a reasonably clear set of minimal common design instructions (i.e. rules) and primitive design elements, constant design features that we can find along the history of urbanism in many different contexts and which are taken as design models (e.g. composition axes, visual axes, orthogonal grids, main squares, etc, which in some cases even take specific names that carry a particular urban meaning such as the *cardus* and the *decumanus*, the *ágora*, the *bastide* plan, etc). In the end everything seems to deal with networks, building clusters and spaces in between.

These two statements find supporting evidence in theory and practice to support our main hypothesis: if we can find a very generic, widely accepted set of primitive urban design patterns supported by historical evidence, we may be able to develop generative urban design systems that can be used in creative processes by allowing designers to develop up their own personal design style by constraining specific rule parameters from a personal selection of design patterns encoded into discursive grammars.

The concept of design patterns has long been appropriated by the information and computer science community (Gamma et al, 1995) following the original concept defined by Alexander et al in 'A Pattern Language' (1977) to capture recurrent solutions for software design. Our approach follows similar principles by encoding Alexander's pattern concept into discursive grammars (Duarte, 2001) able to generate designs that respond to the pattern's requirements.

The first section of the paper explains the theoretical concepts supporting the research, namely the definition of urban grammars and urban induction patterns and the ontology supporting the design system. The second section describes how case studies were used to extract the design rules to be encoded as design patterns. The next section introduces the implementation platform describing its characteristics and the potential of integrating CAD tools with GIS tools. In the fourth section we explain how urban design moves are implemented as urban induction patterns to generate urban plan representations. The fifth section shows the first results of the implementation. The sixth and last sections discuss the capabilities and the potential of the design system and outlines future goals for the next steps of the implementation.

Background - Urban grammars, generic grammars for urban design

The present research resorted to two different formalisms for the implementation of the generation module. Patterns and shape grammars are combined to generate urban designs with a specific methodological approach. The pattern concept we use is an evolution of Alexander's patterns and Gamma et al (1995) design patterns that we called Urban Induction Patterns (UIP). In our concept, UIPs are urban design moves that urban designers recurrently use in their design process and we provide a generative code to them. Urban induction patterns are codified as discursive grammars (Duarte, 2001).

A discursive grammar is a grammar formalism that allows the search of design solutions within the design domain de□ned by a shape grammar (Stiny and Gips, 1972) by looking for designs that correspond to design speci□cations provided through a description grammar (Stiny, 1981). Heuristics are used to guide the generation towards designs that meet the speci□cations. An urban grammar is a speci□c arrangement of UIPs constructed by a designer. The urban designs are generated from this arrangement of patterns by searching for designs that correspond to the speci□cations set in the program for the urban intervention provided by the formulation module.

As a generic de□nition, an urban induction pattern is a recurrent urban design move encoded into a generic grammar that can be applied to replicate the design move in various contexts and an urban grammar is a speci□c arrangement of UIPs. A complete formal de□nition may be found in (Beirão et al, 2010).

The sets of shapes and labels used by the grammars in the generation module are representations of urban concepts. Shapes are representations of urban objects and the labels describe their attributes. The relational structure between these shapes and labels de□nes the ontology of urban space (Beirão et al, 2009b). The three sub-models of the envisioned urban model--formulation, generation, and evaluation--□nd their common relational protocol in this ontology of city concepts, which is de□ned through □ve top-level object classes sub-divided into several sub-classes. The □ve top-level object classes contain object de□nitions for networks, blocks, zones, landscape and focal points and compile representations that can be linked to GIS representation and available contextual data. The objects in the ontology are described through a list of characteristics arranged in a common format as de□ned in (Beirão et al, 2009a) that describes the urban object and identi□es its position in the ontology. The main idea is that the urban objects used by the generation module contain speci□c characteristics that are able to specify partial information about their role in the design representation and therefore in the urban structure they are representing. For instance, an avenue is a street typology which is represented as an object of type polyline, belongs to a speci□c network or networks, has a speci□c hierarchy, a name (an identi□er) and is built out of a set of components (e.g. sidewalk + bicycle path + 2 car lanes + sidewalk), each component involving speci□c parameters. All these characteristics are organized into an ordered list of characteristics which can be used to retrieve information for the rule application.

Application methodology

We used a total of four urban plans in di□erent contexts and from di□erent authors to encode the design rules of the patterns used by their authors (see Beirão et al, 2009a). Our goal was to separate as much as possible the design concepts of each pattern from the speci□city of each designer's language by de□ning very generic grammars that, once arranged in di□erent sequences and instantiated with di□erent values for the rule parameters, could provide speci□c customized design languages and then customized designs. Furthermore, as a kind of validation test, we tried to □nd other applications of these patterns outside our set of case studies. We encoded the patterns as generic as possible so that they could be common to a great variety of plans. The idea is that UIPs encode a common design procedure or design move independently of the speci□c formalism. The rules in each urban induction pattern are in general de□ned parametrically with a broad application range leaving the decision about speci□c parameter values to the designer. The model encompasses two basic design actions in its methodology: □rst, the designer de□nes a broad urban grammar by selecting a speci□c set of urban induction patterns, and second, the designer □ne-tunes the language by setting the parameter values that are still to be de□ned in the patterns. These two actions can be progressively performed along the design process allowing an interactive approach to design and also a progressive and interactive synthesis of the grammar.

Implementation principles

The goal of this implementation is to show that:

1. by using urban induction patterns we can develop a very large scope of urban design languages which cover a very broad variety of design solutions, thus creating the basis for a generative design tool;
2. by using urban grammars the predeﬁned structure of the representation and data components guarantees that the generated designs are drawn in a format that can be immediately exported to the GIS environment in order to be analyzed and evaluated by the available tools and routines.

In this paper we will concentrate our attention on the ﬁrst point.

To illustrate the urban design tool as suggested in the ﬁrst item, a computational prototype was implemented using the Integrated Development Environment (IDE) of AutoCAD. Visual Basic for Application (VBA) and Visual LISP (VL) were the used programming languages. It was possible to use the AutoCAD Application Programming Interface (API) through the VBAIDE and VLIDE, as well as the graphic editor and the object structure already available in the tool, thereby facilitating the implementation process of the proof-of-concept prototype.

This platform was chosen because it combines CAD features provided by the regular AutoCAD drawing and modeling tools with the GIS features provided by AutoCAD Map along with the terrain modeling tools available in Civil3D. Representations can be accessed in these three workspaces and as long as the generated designs are exported to the Map platform with adequate object representations and following the regular geo-referenciation procedures, the designs and data can be accessed from within the GIS module. The evaluation routines can use most of the available GIS-based assessment tools during the evaluation procedures, as well as other tools provided by the evaluation module.

As the object model of AutoCAD contains pre-deﬁned methods and attributes, object classes that inherited and extended functionalities from the standard API could be implemented. Using such process, our main concern was to associate metadata to the generated objects allowing the prototype to work with a semantic structure rather than just geometric and syntactic structures. Using these abilities, the generated elements in the design contain information about themselves and their relative position in the semantic data structure such as, which class in the ontology they belong to. This feature allows a correct communication between the CAD and the GIS environments. Adding metadata to the AutoCAD objects enables the possibility of using such data for controlling various semantic aspects of the design, for instance, by predeﬁning parametric relations involving not just geometry but also semantic qualities of the objects. In addition, by linking object properties to data on a spreadsheet, makes is possible to extend the control and use of external data during the design process.

To implement the generation module, we followed a well known method in computer science, known as incremental development, which consists in starting from very simple routines and then progressively adding new variables, controls and functions to obtain complex programs.

In the next section, we show the ﬁrst results of the implementation of the generation module. So far, we have developed patterns to generate the compositional axes of the urban plan, patterns to generate grids, and patterns to generate urban blocks. A list of some of these patterns is shown in Table 1. The patterns can only be applied if certain features exist in the workspace. If such features do exist, rules take them as their initial shapes or symbols and are triggered and applied recursively until an end state occurs. The end states are deﬁned within each pattern as a set of conditions that always occur within the application scope of the pattern. An UIP can create new features that might be read by other patterns and therefore starting a new pattern application after a selection is made out of the available

candidate UIPs. The common initial elements in any design derivation are basically of two types: (1) the intervention site, an *AcadPolyline* entity, and (2) a set of urban design references, which are existing line or point entities selected by the designer to inform subsequent design decisions by the UIPs. These references are *AcadEntity*s to which the attribute *Ref* is added so that they can be recognized and used by the first patterns. Next, we show the application of a small set of patterns – *MainAxis*, *OrthogonalAxis*, a grid generation pattern *AddingAxes* and *AddBlocktoCells* – which inserts abstract urban blocks or urban units in the cells generated by the grid of axes. We also show the application of *BlockType* to replace the abstract urban blocks by specific block typologies.

Table 1- Table of urban induction patterns and their application in the case studies. The patterns in bold are the ones that have already been implemented.

Urban Induction Patterns ▼ Case studies ►		CG – Praia	CG – Moita	FD, FC, TS – Ijburg	P – Ypenburg	Cerdá – Barcelona
Creating the composition guidelines						
MainAxis	**MainAxistheLongerLine**		●	●	●	●
	Cardus	●				
	MIAxis (Most Important Axis)					
OrthogonalAxis (Decumanus)		● (5×)	●	●	●	●
CompositionAxis					●	●
Creating grids (completing the street network)						
(grid by) **AddingAxes**		●	●	●		●
(grid by) AddingBlockCells (Beirão et al, 2010)		●	●			●
Transformations on street network						
AxisOverGrid					●	●
MoveGridNode				●	●	●
Creating Public Space						
GeneratePlaza		●	●			
InsertFocalPoint		●	●			
Square	SquarefromBlockSubtraction	●	●	●	●	●
	SquarefromBlockTrim	●	●	●	●	●
	SquarefromCornerTrim	●	●		●	●
	TrimPublicSpaceinBlock			●	●	●
Creating Urban Units						
AddBlocktoCells (Beirão et al, 2009b)		●	●	●	●	●
AdjustingBlockCells (Beirão et al, 2009b)		●	●	●	●	●
ClassifyUUnitCells		●	●	●	●	●
DeneUUnit	**BlockType**	●	●	●	●	●
	Cluster			●	●	
InitialUUnit				●	●	●
AddUUnitbyLabel	**AddBlockType**	●	●	●		●
	AddCluster			●	●	
InsertPublicBuilding (service buildings)		●	●		●	●
InsertBuilding (inserts a building in a focal point)		●	●			●
Others – detailing patterns and assigning functions						
AddLocalStreet					●	
AssignFunctions		●	●	●	●	●
ManageBuildingHeight		●	●	●	●	●

Preliminary implementation results

The first step in the design process is provided by the formulation module and consists of the analysis of the design context, extracting and analyzing the available data in order to produce a list of specifications--the urban program--to be taken into account by the generation module However, it might be difficult or virtually impossible to know in advance all the inputs needed for the generation. Some inputs might depend on the reaction to what is being generated. As such, the urban program might be incomplete and might be continuously refined and updated. Considering the possibility of the urban program to be ill-defined or incomplete, the generation module is being implemented in such way as to request data each time it is needed and can not be found in the available specifications. In the current prototype implementation, information might be requested to the user (the designer) whom should be able to fill in the gaps in data as needed. In addition, although most of the information manipulated in the urban design process is objective and therefore can be processed automatically, other information might be subjective and concern stakeholder's expectations or idiosyncratic design options. Therefore, the implementation being developed is open and interactive to accommodate such nuances.

Patterns available for use in the design depend on the progress of the design itself. The first patterns available for application react only to two kinds of initial shapes, the intervention site limit (I_s) and elements selected by the designer as design references (R_{ef}). These references can be any elements of the design context selected by the designer, but the current version of the implementation considers only points in the design context marked by the designer to identify design references. By connecting these reference points the system provides directions that can be used to trace compositional axes. When a pattern is applied, the result may turn existing patterns off and turn additional ones on. The available pool of patterns is, thus, constantly being updated.

Because only a few patterns were implemented so far, design options are mainly limited to the exploration of parametric variations within each pattern. However, a larger set of patterns will enlarge design options to encompass quite other morphological domains. At the moment, this is only perceptible at the level of block generation.

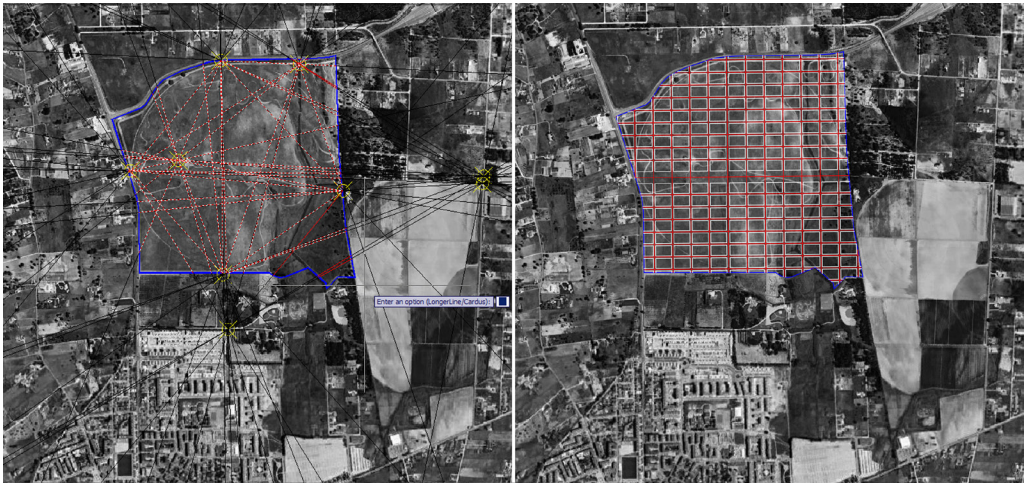

Figure 1 - Grid generation: 1- Selection of points defining reference points; 2- Generation of the grid.

Figure 1 - Grid generation: 1- Selection of points defining reference points; 2- Generation of the grid.1 shows the application of the following sequence of patterns: *Cardus* (actually an option of *MainAxis*) + *OrthogonalAxis* + *AddingAxes* (grid definition by adding axis) + *AddBlocktoCells* (adds block cells in the cells defined by the grid taking into account the street width stored in street representations.

In brief, these urban induction patterns have the following generative behavior:

MainAxis - calculates all the possible axes inside I_s that can be traced among all the selected R_{ef} points and chooses one out of two optional criteria: the *LongerLine* following Frits Palmboom's (author of the Ypenburg plan) expression about his design methods - *"I always look for the longer line in the territory"* - and the *Cardus*, which chooses from the candidate lines the one closer to the north-south orientation.

OrthogonalAxis - calculates all the possible axes that are perpendicular to a selected axis and pass through a R_{ef} point and then selects one out of two optional criteria: *LongerLine* and *MidPoint* which corresponds to the axis that crosses on a point closer to the mid point of the selected axis.

AddingAxes - generates a grid of streets and blocks considering a user defined block dimension. The implementation follows the rules for this pattern shown in Beirão et al (2009). All the generated axes contain data regarding their hierarchical level (out of 4 hierarchical levels, a_1, a_2, a_3 or a_4) to which will correspond user defined street widths.

Figure 2 - AddBlocktoCells - this pattern replaces predefined block typologies in a grid of blocks orienting the front side of the block towards the main street. The 4 user defined block types can be seen on the right side of the image.2 shows the application of a sequence of patterns to generate block typologies starting from a predefined orthogonal grid composed of a random number of streets (between 4 and 8 in each direction) with a random distribution of four hierarchical levels of streets expressed in street width, in which block cell dimensions vary randomly within a predefined interval. Although grid parameters result from designer's input, its purpose is to set an irregular basis for the following patterns: assign different classifications to block cells or abstract blocks (pattern: *ClassifyUUnitCells*), define a user defined number of block types (pattern: *DefineUUnits*, option *BlockType*), and replace the abstract block cells by the types that correspond to the

Figure 2 - AddBlocktoCells - this pattern replaces predefined block typologies in a grid of blocks orienting the front side of the block towards the main street. The 4 user defined block types can be seen on the right side of the image.

predefined block classifications, adapting each type to the existing block dimensions (pattern: *AddUUnitbyLabel*, option *AddBlockType*).

These patterns execute the following procedures:

ClassifyUUnitCells - the user defines how many hierarchies to assign to the cells or abstract blocks in the grid and which blocks have each hierarchy. This selection stores in the AutoCAD entities a string identifying the classification.

DefineUUnits is supposed to allow the definition of urban types of different kinds (blocks, clusters of buildings and neighborhoods) but *BlockType* is the only one implemented so far. *BlockType* defines all the characteristics of a specific block type. The user defines the block by combining two basic typologies out of a set of five (abstract block, closed block, linear block, punctual block and matrix block) through four available operations (subtraction, addition, product and symmetric difference) (Stouffs, 1994). In principle, the designer defines the same number of types as the number of hierarchies assigned to grid cells.

AddBlockType replaces the abstract blocks in the grid by the corresponding block types, adapts the typology to the block geometry and rotates it to match the front side of the block to the higher street hierarchy of the streets surrounding the block. The pattern contains several routines constraining the block structure and building sizes within reasonable and acceptable standards. Although information regarding block types may be defined by the formulation module, their defining variables may be accessible by the designer and are, therefore, customizable.

Discussion and future work

The computer implementation of the City Induction generation module has confirmed that urban patterns have a very complex structure, which constitutes a challenge. As observed in Alexander's pattern language, many patterns are embedded in other patterns and every pattern calls for other patterns. Similar situations occur in this implementation. For instance, some design actions have consequences on other levels of the design involving other components of the urban ontology. While designing, defining a specific distribution of buildings in a block presupposes specific plot subdivisions and vice-versa. Therefore, a pattern for generating buildings should call and constrain the rules of patterns for generating plots and vice-versa. Although the previous development of the urban space ontology helped to clarify the relations among the various patterns, the computer implementation permitted to gain additional awareness on how to relate and build up these patterns. This constitutes an important step for future work.

Developing an implementation is in itself an act of designing. In the case of a design tool, this means that one is confronted with different possibilities regarding which aspects to leave for user decision and which aspects to automate. An extensive list of possible improvements became clear during the work; some were already in the agenda. Here follow some examples of planned future work:

- Defining references to stimulate design options is a very important aspect of the design process and it is not just related with the position of the elements but also with its meaning and shape. In order to improve this aspect we are planning to (1) add weights to the Ref points, (2) create linear references and (3) relate references to meanings (e.g. distinguish references like historical landmarks from natural features).
- Develop some automated approaches to the distribution of the block hierarchy according to urban design data and criteria such as density, street hierarchy, and weighted references which, in turn, will require the development of a multicriteria weighted approach.
- Automate plot subdivision following block design decisions.
- To store predefined types to reuse or transform in other design contexts. In this case, it will be useful to store also information regarding the context in which they apply so that they are made available only when such context occurs.

Conclusion

Design can be atomized into small design acts or moves which can be encoded into individual generative codes that reproduce the results of such design moves. The codes are very generic discursive grammars which allow for a very broad interpretation of the meaning of these design moves. Particular customization of a design move is made available through the optional sub-patterns and by constraining the available parameters. Although still very limited in the available prototype, what builds up a designer's language is the choice of patterns s/he makes along the design process and how s/he interprets the patterns, i.e., how s/he constrains the parameter values in the pattern's rules. Future implementation of additional induction patterns will enlarge the scope of the corresponding generic grammars, thereby enlarging the design space of the tool being developed. Then, it might be generic enough to become an interactive and intelligent tool, to use in creative design.

The City Induction system's architecture de□ned by the urban ontology and the pattern-encoding grammar formalisms produces structured representations that constituted the basis for a city information model. In addition, it facilitates the communication between the GIS environment providing contextual information and the CAD environment in which the generation of designs occurs, which then feeds back information regarding the designs into the GIS.

Finally, AutoCAD provides for interfaces that may be used to develop customized CAD objects and de□ne routines encoding urban induction patterns. This permits designers to develop personal design languages and bene□t from the advantages of using generative rules in design exploration.

Acknowledgements

The City Induction project is funded by 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 FCT, grant SFRH/BD/39034/2007. Gelly Mendes is funded by FCT through the City Induction project. 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 for their comments and support.

References

Alexander, C, Ishikawa S and Silverstein M 1977, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, New York.

Beirão, J, Duarte J and Stou□s R 2009a, 'Grammars of designs and grammars for designing' in T. Tida□ and T. Dorta (eds) *Joining Languages, Cultures and Visions: CAADFutures 2009*, PUM Canada.

Beirão, J, Duarte J and Stou□s R 2009b, 'An Urban Grammar for Praia: Towards Generic Shape Grammars for Urban Design', *Computation: The New Realm of Architectural Design 27th eCAADe Conference Proceedings*, Istanbul, Turkey, pp. 575-584.

Beirão, J, Montenegro N, Gil J, Duarte J and Stou□s R, 2009c, 'The city as a street system: A street description for a city ontology', *SIGraDi 2009 - Proceedings of the 13th Congress*, São Paulo, Brazil, pp. 132-134.

Beirão J, Duarte J and Stou□s R 2010, 'Creating Speci□c Grammars from Generic Grammars: Towards □exible Urban Design', *Nexus 2010: Architecture and Mathematics, Conference edition of the Nexus Network Journal*, Porto (forthcoming).

Duarte, J 2001, *Customizing Mass Housing: a discursive grammar for Siza´s Malagueira house*, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge.

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

Knight T 1981, 'Languages of designs: from known to new'. *Environment and Planning B: Planning and Design*, 8, pp. 213 – 238.

Knight T 1983, 'Transformation of languages of designs: part 3', *Environment and Planning B: Planning and Design*, 10, pp. 155 – 177.

Schon, D 1983, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

Stiny, G and Gips J 1972, 'Shape Grammars and the Generative Speci□cation of Painting and Sculpture'. *Information Processing*, 71, 1460-1465.

Stiny, G 1981, 'A note on the description of designs', *Environment and Planning B: Planning and Design*, 8, 257-267.

Stou□s, R 1994, *The algebra of shapes*, Ph.D. dissertation, Pittsburgh: Department of Architecture, Carnegie Mellon University.