*Artificial Intelligence for Engineering Design, Analysis and Manufacturing*

**cambridge.org/aie**

# Generic grammars for design domains

José Beirão[1] and José P. Duarte[2]

[1]CIAUD, Faculdade de Arquitetura, Universidade de Lisboa, R. Sá Nogueira, 1349-063 Lisboa, Portugal and [2]SCDC, Stuckeman School of Architecture and Landscape Architecture, Penn State University, 150 Stuckeman Family Building, University Park, PA 16802, USA

**Abstract**

Shape grammars have been developed to codify a specific type of artifact – Queen Anne houses, Buffalo bungalows – or the style of a particular designer – Andrea Palladio, Frank Lloyd Wright, or Álvaro Siza Vieira. However, these specific grammars fail to encode recurrent design moves or features that are above the particularities of a specific design style or the idiosyncrasies of a specific designer and, therefore, are common to a larger category of designs and maybe reutilized and incorporated in the definition of new, specific design languages. To overcome these limitations, the notion of generic grammars for defining design domains is introduced. Its application to the urban design domain is illustrated by showing a generic grammar implementation resulting in a City Information Modeling platform composed of a parametric design interface connected to a geographic database.

**Research Paper**

**Cite this article:** Beirão J, Duarte JP (2018). Generic grammars for design domains. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **32**, 225–239. https://doi.org/10.1017/S0890060417000452

Received: 25 October 2017
Revised: 19 June 2017
Accepted: 20 June 2017

**Key words:**
CIM; generic grammars; parametric design; shape grammars; urban design

**Author for correspondence:**
José P Duarte, SCDC, Stuckeman School of Architecture and Landscape Architecture, Penn State University, 150 Stuckeman Family Building, University Park, PA 16802, USA.
E-mail: jxp400@psu.edu

## Introduction

In recent years, several researchers have addressed the development of generic grammars in different fields of design, including urban design (Beirão et al., 2010), the design of public spaces in social housing developments (Mendes et al., 2013), housing design (Benrós et al., 2014), chair design (Garcia & Barros, 2015), and tableware design (Castro e Costa & Duarte, 2013). In these works, generic grammars are considered to define meta-design languages that can be applied generically in a design domain independently of local specifications or particular idiosyncrasies of a specific design language. By constraining the domain application of a generic grammar, the grammar becomes specific, meaning either style-specific, context-specific, customized, or any combinations of these. The purpose of this paper is to provide a formal definition for generic grammars and show their use and utility to generate designs in a particular design domain.

Historically, the use of rule-based systems have been considered appropriate for four different levels of use: (1) at an analytical level as a way of understanding design rules, design processes and historic settlements within a design domain or a design language, such as in (Stiny & Mitchell, 1978; Koning & Eizenberg, 1981), to name some early examples; (2) as synthesis tools that can be used in the generation of designs (Knight, 1992); (3) as regulations or regulatory devices controlling the generation of new instances to be kept within the limitations of some domain (or sub-domain) (Lehnerer, 2009; Marshall, 2011); and (4) as predictive devices in the development of simulation algorithms with the aim of simulating the effects of hypothetical sets of rules in particular contexts (Shea & Cagan, 1997; Caldas & Norford, 2002).

In this paper, we introduce the concept of "Generic Grammar" as a set of production systems (Gips & Stiny, 1980) acting at the four aforementioned levels, using rules as the generative element and operating within the particular frame of a well-defined design domain. The concept of "Generic Grammar" is analytical to the extent that any generic grammar is inferred from specific grammars that resulted from the analysis of different corpi in the same design domain. It is synthetic because it may be used to define specific grammars that differ from those used in the inference. It is regulatory as the resulting specific grammars may be used to define new objects in the same design domain, thereby implicitly setting the boundaries of the domain. Finally, it is predictive because the resulting specific grammars can generate designs that match predefined criteria or performance, according to the given viewpoints. Below we show an application of the concept to the urban design domain, which can be applied to different design domains, as shown by the works by Castro e Costa and Duarte (2013), Benrós et al. (2014), and Garcia and Barros (2015).

The paper is divided into eight main sections. Section 2 addresses the concept of "Design Domain," showing how it can be defined with enough accuracy using an ontological approach complemented by a production system. Section 3 reviews the concept of production system focusing on patterns and shape grammars and postulates some basic notions on how various grammars can be correlated through formal relationships defined within a design domain to develop generic grammars. As a way of illustrating the proposed conceptual and methodological framework, Section 4 presents a simplified version of a generic grammar for the urban

© Cambridge University Press 2018

CAMBRIDGE UNIVERSITY PRESS

design domain, and Section 5 shows its implementation as a parametric design platform and its application to a particular urban context. Section 6 provides a formal definition of generic grammars for specific design domains, and Section 7 discusses methodological issues to take into account in the implementation of generic grammars. The paper ends with a concluding section that summarizes these issues and draws some concluding remarks regarding the purpose of generic grammars.

## Design domains, ontologies, and generic grammars

The term "design domain" refers to a formal or conceptual field that may be the subject of a design activity, such as urban design, chair design, or bottle design. The main difficulty, however, lies in the formal definition of a design domain. Such definition should be able to clarify which are the objects of a design domain, classify them correctly by clearly stating their types, parts, and relations. This is important to enable a structured look into the morphological and topological aspects of the domain, and to develop rules acting on such objects and relations, either symbolic relations or spatial relations.

Such a structured description of a design domain can be accomplished by characterizing the domain's ontology. In computer science and specifically in the field of knowledge representation, ontology is the set of specifications of a domain's conceptualization (Gruber, 1993). More accurately, ontology describes a domain in terms of the "objects" (or individuals) of which it is composed, object arrangements in terms of a structured classification identifying the "classes" and "subclasses" of objects composing the domain (taxonomy), their "attributes," and the expressed "relationships" among them (topology).

"Objects" or individuals are singularly identifiable entities that may be instantiated or manipulated as such. In a shape grammar-based (Stiny, 1980) representational structure, objects may be represented by labeled shapes which typically have a geometric component (a shape $S$) and a symbolic component (a label $L$). The shape part or the label part of a labeled shape may be empty. Labels without a shape part are simply abstract classifiers or symbols, concepts that may or may not be related with shapes. Unlabeled shapes are abstract shapes without semantic value other than their geometric properties. Labeled shapes are geometric entities attached to a concept and therefore we say they are semantically meaningful shapes. In addition, in the case of more complex objects, labeled shapes may be articulated and complemented by descriptions, adding other layers of information (Stiny, 1981).

"Classes" are types of objects organized in terms of a dependency structure based on object typology. Objects with shared characteristics are organized into common classes. Complex objects in a class may be composed of simpler objects belonging to a subclass or several subclasses thereby defining dependency relationships. Objects which are not composed of other objects are called primitives. Furthermore, objects in classes can be represented by shapes, labels, or labeled shapes depending on the kind of representational criteria used for classification in the ontology structure. Hence, a class $C$ is represented by a vocabulary of shapes $S$ and labels $L$. Formally,

$$C_i = \{S_i, L_i\} \tag{1}$$

where $i$ is a class identifier index.

"Attributes" are characteristics, properties, or parameters of objects in a class. To avoid confusion, note that attributes correspond to properties that may be common to objects in several classes of the ontology (e.g., point coordinates), while labels are specific of a particular class in ontology (e.g., positions of protected arboreal species may be given by attaching a label or a symbol representing a particular tree species to a point in the right location).

"Relationships" express the dependency structure between objects and consequently also between classes.

In geometrical terms, primitives are points, lines, surfaces, and solids, with zero, one, two, or three dimensions, respectively. Each of these primitives is bounded by primitives of the dimension immediately below. As such, solids are bounded by surfaces, surfaces by lines, and lines by points. Points have no boundary, so, at the end, all primitives are defined by points, considered the primal of the primitives. The main attribute of primitives is their positions in the design space, defined by coordinates in a given reference system.

In the "urban design domain," the ultimate object is the city, its parts, and the relations among parts. This structure should contain all the concepts needed to describe cities as they are and cities as we intend them to be, in other words, describe the transformations that may be applied or that we admit possible within a city while following some goal, which we will call "development vision." The term "urban design" stresses that the domain describes not just the urban space "as it is," but also "as it is planned or envisioned to be," including the actions and processes that acting upon existing objects transform them into new objects according to established planning rules. Figure 1 shows a classification structure developed for the domain of urban design. The larger object – the city – is divided into five main groups of object classes, which are called systems. These systems represent different ways of seeing the city and contain the main urban elements, which we are able to deal with, more or less independently from one another: (1) the city seen as a street system; (2) the city as a built system; (3) the city as a property system; (4) the city as a physical system; and (5) the city as a system of focal points. These systems can be independent topics of analysis involving subjects that are usually addressed in a reasonably autonomous fashion, such as network analysis (Hillier et al., 1987; Marshall, 2005) and urban morphology analysis (Muratori, 1967; Coelho et al., 2013), to list the most common. Each system is composed of object classes organized according to their expressed relationships. For instance, considering the street (or public open space) system, we have as a top class axial representation of the network (AN) followed by classes which are pure classifiers of streets: transportation network (TN) and street nomenclature (SN). We call pure classifiers to classes with an empty shape set, containing only labels which describe concepts that should be classified independently, even though such concepts may be attributed to shapes by means of rules. For instance, the class TN represents street classification seen for their role in the TN, and we may find class object instances such as high-speed roads, distribution streets or, more specifically, local distribution or local access streets (Fig. 2). The class SN represents street types as usually identified in common languages,[1] such as

[1]Note that this class is language dependent from the cultural context and, therefore, should be adapted accordingly. The same can be said of TN because transportation types are usually dependent on local planning definitions. However, in the latter case it would be easier to establish a widely accepted standard. It can also be added that language brings an intense cultural value into the ontology.

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

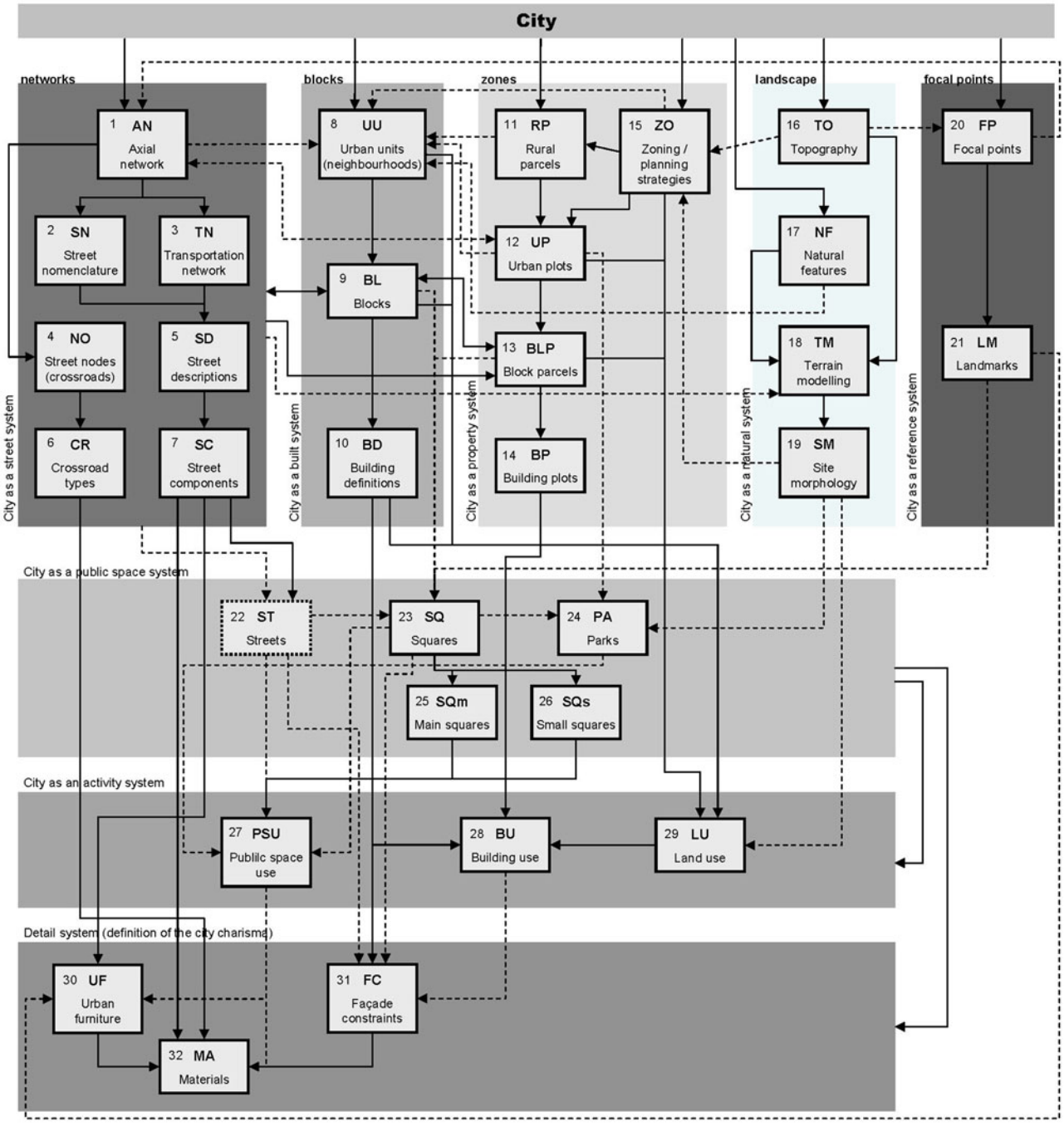



**Fig. 1.** Ontology proposed for the urban design environment, which is one among many that can possibly be developed for this domain. Primary relationships are indicated by a continuous line. Secondary relationships are represented by a dashed line.

"alley", "boulevard", and so on (Fig. 3). Each street type in TN and SN classes has a street description (SD) consisting of the minimum set of street components (SC) indicating the number of car lanes, sidewalk areas, or other components of a street section. In other words, this is a formal description of a street section. The street section may be enlarged according to permissions set in terms of vicinity rules between SCs and within a variation range established for each parameter in a SC (Fig. 4). For instance, "canal" (⑩) is included in the set to cover streets in The Netherlands, where canals may participate in the definition of street profiles.

The structure explained above defines relationships among all objects in the ontology and consequently also relationships among classes. The relational structure between objects and classes predefines the way objects may be used in design rules.

## Production systems, patterns and grammars

Gips and Stiny (1980) proposed a uniform characterization of Post's production systems (Post, 1943) by underlining that their common structure was composed of: (1) the objects they process, (2) their definitions, (3) their interpretative mechanism, and (4)

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

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

**Fig. 2.** Generic grammar for urban design: transportation network (TN) and street descriptions (SD) for selected street types within TN. This ontology allows for variations and comparison with Marshall's classification in terms of *stratification by speed* and *connectivity route types* according to structural role of streets within the network.

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

**Fig. 3.** Generic grammar for urban design: street nomenclature (SN) and street descriptions (SD) for selected street types in SN, and its comparison with Marshall's route classification in terms of *stratification by speed* and *connectivity route types* according to their structural role in the network.

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

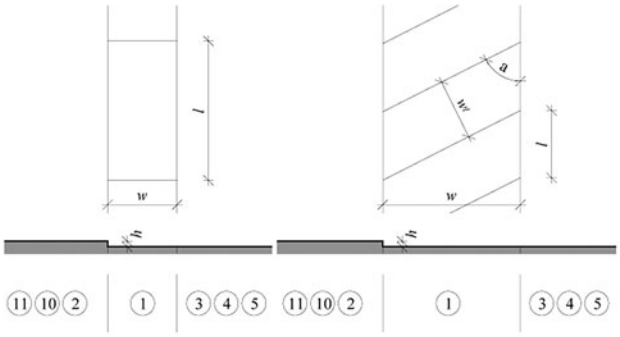
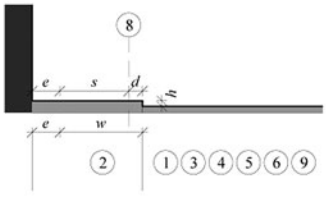
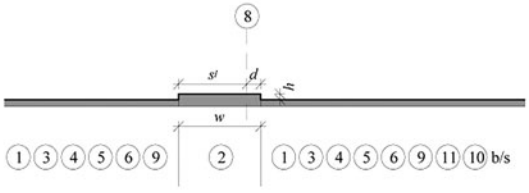
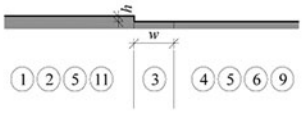
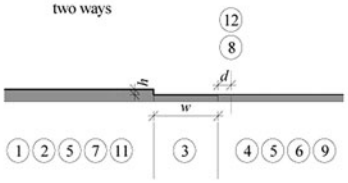
| Street Components (**SC**) – a collection of street profiles | Profile schema – indicates profile parameters and possible adjacent profiles | Profile parameters |
|---|---|---|
| ① - street parking<br><br>Valid for<br>S3 – S2<br><br>Preferred for<br>S2.5 – S2 | Parallel<br><br><br>Not parallel<br><br> | Parallel<br>$2.0 \le w \le 2.4$<br>$4.7 \le l \le 5.5$<br><br>Not Parallel<br>$4.8 \le w \le 5.5$<br>$2.25 \le l \le 2.5$<br>$45° \le \alpha \le 90°$<br><br>$0 \le h \le 0.2$<br><br>If ⑩ $h = 0$ |
| ② - sidewalks<br><br>S1 | <br><br>Central sidewalks<br><br> | $1.25 \le w \le 5.0$<br>$s \ge 1.2$<br>$w = s$ + ⑧*width* + $d$<br>$0.3 \le d \le 0.75$<br>$e$ is an extra space for additional purposes. E.g. – esplanade, benches, telephone booth, commercial activities, etc.<br>$0 \le e \le 2.5$<br>And can be used also as tolerance<br>The $w$ value is further restricted depending on the street type to which it belongs. The values indicated here encompass all possiblities in the case of central sidewalks.<br>For central sidewalks –<br>$1.25 \le w \le 2.0$<br>$w = s'$ + ⑧*width* + $d$<br>$0 \le h \le 0.2$<br>$h = 0$ in the case of ⑥, ⑩ or ⑪ |
| ③ - bicycle lanes<br><br>S1.5 | one way<br><br><br>two ways<br><br> | One way<br><br>$1.0 \le w \le 1.5$<br><br>Two ways<br><br>$2.5 \le w \le 3.0$<br><br>Independent<br>(not next to ④, ⑤ or ⑨)<br><br>$2.0 \le w \le 3.0$<br>⑫ in the case of **R1** apply for only with ⑥ |

**Fig. 4.** Generic grammar for urban design: street components (SC). This table shows the 12 street components that once combined allow for the generation of almost any street type within a city network (only three are shown). The table includes details of the variable parameters and rules to combine them according to the permissions set as street descriptions (SD) – see the two previous tables.

the objects they generate. Through this uniform characterization we can see most algorithmic structures as production systems defined by rules with the format $u \rightarrow v$, where $u$ and $v$ are objects from a uniform class $C$ of well-defined objects, and $C$ may be seen as any subclass of labeled shapes of the form $C_i = \{S_i, L_i\}$ in an ontology defining a particular design domain or part of a design domain (sub-ontology). In such cases, the production systems are shape grammars (Stiny & Gips, 1972) operating with shapes

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

belonging to a particular class of well-defined objects in a design domain. In fact, they are discursive grammars (Duarte, 2005) as we will explain below. Technically, a discursive grammar consists of a shape grammar, a description grammar (Stiny, 1981), and a set of heuristics, which are used to guide design generation toward designs with desired properties, by comparing the description of the evolving design with the description of the desired design, and then deciding which rule to apply next.

Another form of algorithmic structure was introduced in the late 1970s by Alexander et al. (1977) through the concept of a pattern language. At an abstract level, each pattern is composed of a predicate condition for which a set of generic time-tested solutions may be applied as a consequent action. The predicate condition is also said to correspond to recurrent phenomena in a design domain (urban or architectural design), hence implying that typical design problems within these design domains may have already typical, generic time-tested solutions. In this definition, the form $u \rightarrow v$ is transformed into a more ambiguous format, predicate → consequent, where $u$ and $v$ become more complex concepts involving some, also complex, set of transformations of the predicate conditional state into a consequent state. Patterns in this context describe generic design operations and designs are arguably the results of the application and instantiation of particular sequences of patterns. Sequences are not predefined but desirable relations with other patterns are described, both in the predicate and in the consequent parts of each pattern, thereby reinforcing the algorithmic structure of the pattern language.

Such particular forms of algorithmic design, patterns and shape grammars are said to encode languages of design because they define formally the syntactic rules of a particular design space in a design domain. These rules can be used to generate designs and the set of all the designs that can be generated from such rules form a design language. Both forms have their strengths and weaknesses. The accuracy and focus on syntactic rules in shape grammars (such as in linguistics) raises many issues regarding semantics (Fleisher, 1992) as well as difficulties in computer implementation (Gips, 1999), while the vague descriptions of patterns lack accurate formalization but are open to semantic interpretation and to varied forms of implementation. Two particular aspects of patterns may be underlined. First, the computer implementation of a pattern can be often defined by a shape grammar and, surely, by some form of the production system, although such a production system may need a rather complex interpretative mechanism to encode all the semantic subtleties of the pattern. Second, semantics in a pattern language is also guaranteed by the relational structure defined among patterns. We may say that there is a specific ontological structure underlying a particular pattern language. As a result of the preceding observations, we propose that a generic grammar is composed of a set of different kinds of production systems operating on objects from an ontology describing a particular design domain. In the following sections, we will illustrate the concept of generic grammar by describing one for the domain of urban design.

## Generic grammars for the urban design domain

In 2012, Duarte and Beirão (2012) described an algorithmic approach to urban design that used Alexander's et al. (1977) Pattern Language and Stiny and Gips' (1972) shape grammars that they had been using in teaching and in practice. The basic idea was that following contextual analysis, certain patterns would be triggered, forming a program for urban intervention, and then such patterns could be formalized in a flexible urban plan using shape grammars. Later on, in "City Induction" Duarte et al. (2012) formalized the underlying methodology with the aim of developing a supporting computer platform. In addition to the use of patterns and grammars, it was proposed the development of an ontological structure (Gruber, 1993) to describe the urban environment and the development processes, and space syntax (Hillier & Hanson, 1984) to characterize affordances of urban solutions.

In CItyMaker (Beirão, 2012), Beirão defines a common structure for Urban Induction Patterns (UIPs). UIPs are presented as algorithms for urban design involving a structure inspired by Gamma et al. (1995) that include shape grammars as template codes for describing typical urban design operations. Inspired by Alexander and more closely by Gamma et al. (1995), an UIP is composed by the following parts: "*Name/Intent/Also Known As/Known Uses/Description/Structure/Predicate/Consequent/Discursive Grammar/Related Patterns*" (Beirão, 2012, Appendix 2 – A Library of Urban Induction Patterns).

*Name/Intent/Also Known As* correspond to Gamma's definitions. *Name* states the pattern's algorithmic behavior in a reasonably self-explanatory short statement; *Intent* gives a brief explanation of the pattern's algorithmic principles; and *Also Known As* gives an alternative name also short and self-explanatory. *Known Uses* illustrates the algorithm's application in a way similar to Alexander's "archetypal illustration" including an additional explanatory statement or summary. *Description* gives a simplified verbal description of the pattern's algorithm. *Structure* provides a visual relation of the object classes used by the algorithm and a short graphic pseudo-code stressing the algorithm's main purpose. *Predicate* describes the underlying conditions upon which the pattern can be applied. In other words, it accurately describes the initial conditions for the application of the UIP, giving information on which class $C_i$ to find the initial shapes and labels upon which rules may be applied – the matching conditions. *Consequent* gives a description of the expected results. Gamma's sample codes are replaced in this structure by a *Discursive Grammar*, which formally gives the rules regulating the generative behavior of the pattern – the design action itself. *Related Patterns* work much in the same way as in Alexander et al. (1977) and Gamma et al. (1995), but in this case the state-specific relationships between object classes or, in other words, to what object classes can a pattern be applied and what patterns may be applied to objects resulting from a pattern's application. These are, in fact, general matching conditions stated at the class level.

By using this formalism, CItyMaker presents a generic grammar for designing urban plans. The author argues that this generic grammar is composed of 45 UIPs that combined in different ways can produce different urban plans involving different languages of design. Each pattern can be constrained within the set of options defined by its rules and also within the variation range of their parameters. It can be argued that this set of patterns constitutes a pattern language involving a wide spectrum of design solutions approximately in the same way as proposed by Alexander et al. (1977). A specific urban design language may be obtained by choosing particular arrangements of patterns. A specific design instance may be obtained by choosing from this specific urban design language, a particular rule sequence, and specific values to assign to their parameters.

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

Note that the term "language" has slightly different meanings in the context of Alexander's pattern language and Stiny's shape grammars, but they both hold and are related in our concept of generic grammars. For Alexander, a set of patterns forms a language that can be used to specify the ingredients of designs that are appropriate for given contexts whose features trigger just certain patterns. Once made, a pattern is triggered and included in the design brief, it may be instantiated in the design. For Stiny (1980), the set of designs that can be generated following the rules of a given shape grammar form a design language. We manipulate the pattern language associated with a generic grammar to define a less generic pattern language by choosing to use just some patterns. Then, we use this sub-pattern language to formulate the design brief. Then, we manipulate the generic sub-grammars associated with the selected patterns, by selecting some rules and constraining rules' parameters to define specific sub-grammars. Finally, we use these specific grammars to derive the design. These steps do not necessarily need to be performed in this rigid linear order, as designing might imply to proceed back and forth, until the problem, the solution, and the language are defined. The meta-language for designing defined by a generic grammar defines a very large design language formed by all the designs that can be generated by all the sub-pattern languages and all the specific grammars that can be defined from the generic grammar.

As a way of illustration, let us consider just a few patterns and some basic urban design operations. In a urban design, a designer needs to pass through a long sequence of design decisions. Roughly, designers follow four different levels of design decision which may or may not be applied consecutively (Beirão & Duarte, 2009) but are somehow scale dependent: (1) the wider level involves rules defining the plan guidelines through broad compositional guidelines that anchor together some significant landmarks in the territory; (2) another level lays down the grids ruled by the compositional guidelines; (3) the next level defines rules for defining urban units, such as neighborhood, urban blocks, and urban plots, together with minimum program requirements of neighborhoods regarding public facilities and open spaces; (4) the last level involves rules for qualifying the urban environment, such as public space design and façade design constraints, including material aspects in both cases, among other aspects. If we look simply at the street layout without much consideration for other aspects of the design, we will be applying rules involving objects taken from the "network" part of the ontology (Fig. 1). Therefore, rules will operate on objects from classes AN, SN, TN, NO, SD, CR, and SC. Rules are arranged in the form of UIPs with the structure and algorithmic description found in Table 6 in CItyMaker (Beirão, 2012, pages 117–121). This table contains a structure that embeds the typical procedures designers need to follow to develop an urban plan. The structure can be found in the subsections of the table which give subliminal information on the sequence of decisions that should be taken to design a complete urban plan: (A) creating compositional guidelines, (B) creating grids, (C) street network transformations, (D) creating public spaces, (E) creating urban units, and (F) detailing spaces. The amount of possible combinations is potentially very large, so developing an example requires reducing the full potential of the grammar. Still, the grammar allows designing from very generic plans to detailed representations of public spaces, depending on which UIPs are used.

In the development of an urban plan, a designer starts from a set of representations depicting preexisting conditions, which usually are available from a geographic information system (GIS). Such representational structure is usually very complex but for the purpose of a design process we will consider it a set of preexisting labeled shapes – the context – represented as set $E$. One may imagine an entire representational structure within the bounds of set $E$ that could be used to trigger different sets of transformation rules based on existing attributes of the context. In fact, the representational structure is that of the GIS system and it provides information regarding the pre-existing environment.

In any kind of urban plan, designers use contextual elements as references for making design decisions. While selecting these elements, according to some criteria, including idiosyncratic intentions, the designer interprets the context and assigns meaning to the place. In our generic grammar, references are used as initial shapes of the design process. The first patterns (UIPs) that can be used are those that contain rules whose left-hand sides match on a reference shape $R_{ef}$. The designer may consider preexisting shapes as erasable shapes (e.g., buildings to be demolished) and label them with $E_{er}$., or as reference shapes, that is shapes to support the initial moves of the design process, and label them with $R_{ef}$. These shapes are so marked by using simple selection methods. Embedded in the selection process there is a classification rule which transforms an object $e \in E$ into either an $E_{er}$ shape or a $R_{ef}$ shape, where $E_{er}$, $R_{ef} \in E$. Additionally, a closed polygon $I$ is set by the user to define the limits of an intervention area. This polygon represents the rule application space or, to be more precise, new shapes can only be generated inside $I$, although rules may act based on references found outside the intervention area. The intervention area $I$ can also be used as an initial shape.

## Implementing and applying a generic grammar for urban design

In this section, we will present the implementation of the simplified grammar presented above into a computer platform, and provide a short example of its application to a possible design situation, starting from a preexisting context where an intervention area $I$ has been defined. The design shown and described on the following pages was generated using the interactive interface available in CItyMaker. This software translates patterns firstly encoded as shape grammars and converts them to parametric design patterns, a necessary step to facilitate computer implementation. This conversion sacrifices embedding, an important property of shape grammars, by which shapes that were not explicitly designed can be identified (e.g., a square resulting from the intersection of two squares). The obtained parametric design patterns are reusable in different types of context and amenable for designers to produce their own solutions. Nonetheless, such a conversion still allows us to demonstrate how the concept of generic grammars provides a valid theoretical framework for structuring the urban design process.

Figure 5 shows the initial steps of a design session using CItyMaker to apply the simplified generic grammar to a particular site. Figure 5a shows only shapes from $E$, the shape set representing the context, and $I$, the intervention area. The area pictured in Figure 5 is an industrial area close to the World Heritage town of Sintra, located some 30 km to the west of Lisbon, Portugal. This industrial area has long been famous for its stone quarries and stone transformation industries. Over time, such activities attracted several other industries, mainly in the construction field. However, the initial rural structure of the territory is still

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

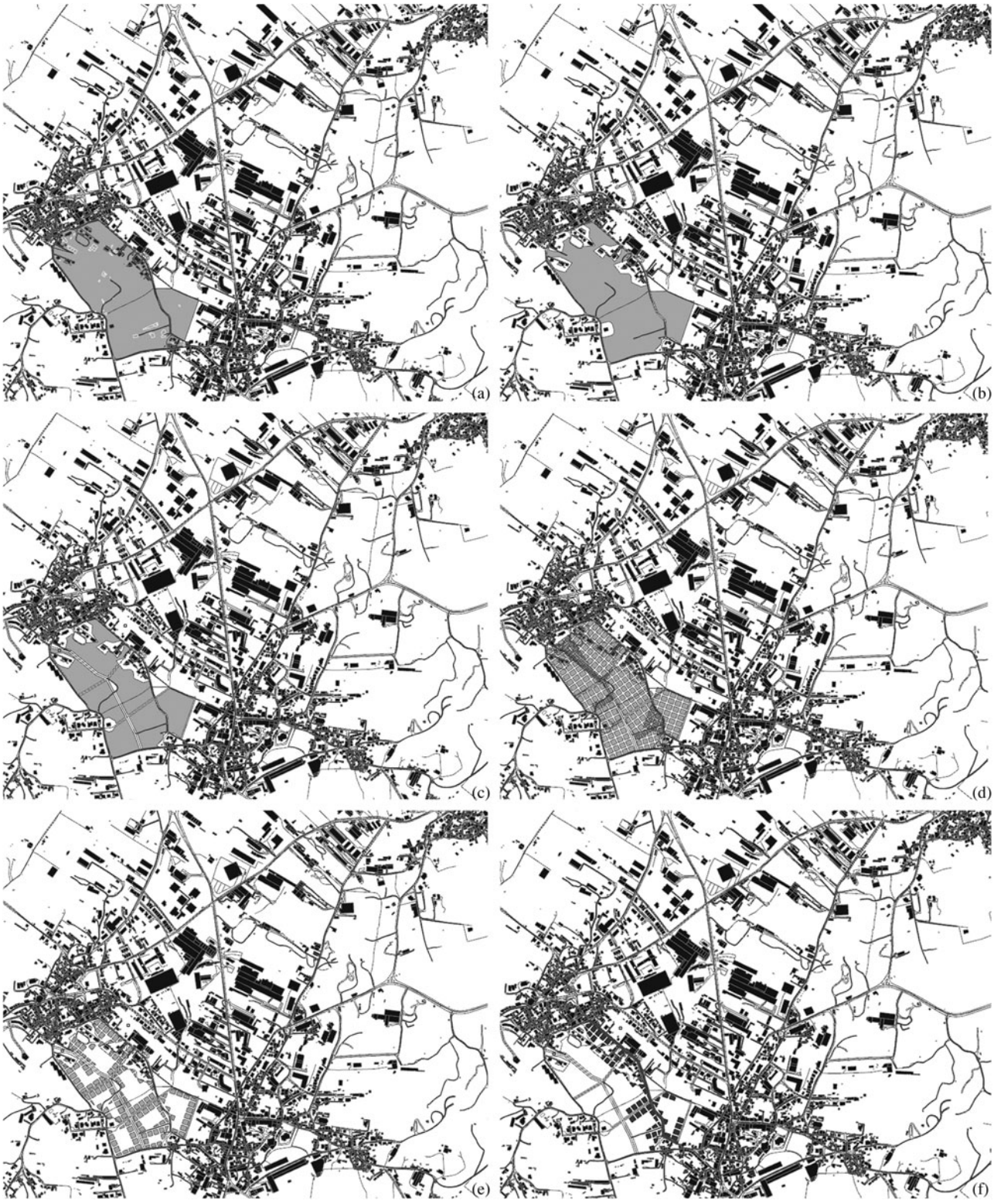



**Fig. 5.** Initial steps of a design session using the computer platform to apply the simplified generic grammar to a particular site: (a) Design context: the intervention area is shown in light gray and buildings that can be erased in dark gray. (b) A "buffer" zone around existing houses to be maintained is subtracted. (c) Addition of new streets defining a grid of "islands" using the *ManualGrid* pattern. (d) Application of a regular grid pattern to each "island." Each grid is rotated according to the longest edge of the corresponding "island." (e) Plots along the streets are retained, whereas others are filtered out. (f) A possible solution following the selection of two village centers and the connecting street as attractors and a growth input expressed in terms of the number of dwellings.

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

patent in the organic tissue of the three village centers contained in the area as well as the regular original *latifundium* structure. Nowadays, these village centers define a polycentric nuclei structure gravitating around the industrial areas. The regular property structure still underlying the industrial areas is not the outcome of a planning operation but simply the result of the occupation on demand of the old rural structure (*latifundia*). However, the occupation of the rural structure by the industry followed no plan and created several quarry spaces resulting in an overall chaotic appearance. At present, the local industry is traversing a period of economic crisis resulting in the abandonment of some quarries and factories. The municipality seeks solutions for these spaces but finds no secure answers due to the crisis and especially to the overall shrinking scenario (population decrease, aging, and investment shortage). Some of the development visions that seem more sustainable gravitate around environmental requalification toward a mixed industrial–recreational park and/or the promotion of housing for the elderly and public facilities, integrated into a large green park.

The criteria used for defining polygon *I* shall not be questioned here, although it could be the result of some sequence of analytical procedures on the context using available GIS information. In fact, it is not the aim of this paper to explain the motivations and principles behind design solutions, nor the results as responses to such principles, but simply to illustrate the operational aspects of the generic grammar and its field of application. The first steps of the design process are a subject of human decision based on the available information or the result of programmatic specifications defined by a programming algorithm (Montenegro et al., 2011). Some existing buildings found inside the intervention area are preserved and others are demolished based on information available in the GIS database (DB). For instance, buildings with a footprint smaller than 20 $m^2$ or in ruinous condition or abandoned factory plants are to be filtered out or erased[2] from set *E* and, therefore, buildings in such conditions will not constitute a limitation for the next design steps. This selection is performed through simple query procedures like those used in common GIS software, here replicated using filter-based design patterns as explained further below. The buildings to integrate into the design define buffer areas, which are subtracted from the intervention area *I* (Fig. 5b). Alternatively, one could simply subtract the corresponding plot areas.

The design described on the following pages was developed in a design environment composed of CAD software and a parametric visual programming interface (VPI). In this case, the CAD software was *Rhinoceros* with the plug-in *Grasshopper* as the VPI. The VPI was extended with the plug-in *Slingshot* to establish communication with a *Postgres* DB, connected to a GIS, and to an urban measures calculator (UMC). The design was obtained using a sequence of parametric design patterns programmed in Grasshopper corresponding to the following descriptions:

[2]In practical terms, 'filtering out' means moving a set of objects from set *E* to one hidden layer allowing the system to keep track of design history. The reader should keep in mind that set *E* is a complex structure of information, also part of the urban design domain ontology. As such, what happens here is the moving of several objects from different classes in the ontology into a unique class of 'demolished' objects represented as labeled shapes, where the label indicates the class to which the 'demolished' object used to belong, hence keeping a track of design history. The term 'erasing' is used here as a simplification, more in tune with what goes on in the designer's mind.

### Geographic information import patterns (GeoInfoPatterns)

These patterns import information and geometry available in a geographic DB. They establish the connection between the DB and the design environment, with the geometry being previewed in *Rhinoceros* and data accessed in *Grasshopper*. They are defined by rules that read information on one site and write information on another, in the same way as rules in parallel grammars. Their role is to make available preexisting shapes, that is, set the context *E*. Each of these patterns is composed of two sub-patterns. The first sub-pattern imports objects' geometry to the VPI. The geometry can then be previewed in the CAD environment. The second sub-pattern imports data related to each object; to be more precise, these sub-patterns are a kind of *QueryPatterns* (see description below), because they have no mapping function. Once objects are imported, the user can select a column of information in the object's table. Then mapping functions may be added to visualize data in the CAD environment much as in a GIS environment.

*ImportBuildings* – As the name suggests, this pattern imports information concerning buildings and as described above it is composed of two sub-patterns. The first sub-pattern imports building geometry to the VPI and the second the data related to each building.

*ImportStatisticSub-Sections* – Imports information provided by the Portuguese National Statistics Institute (INE) organized into statistic sub-sections. Again, the pattern is divided into two parts: one for importing the geometry and the other to associate information regarding each statistic sub-section with the corresponding geometry.

*ImportStreetCenterLines* – Similar functionalities.

*ImportStreetSurfaces* – Similar functionalities.

The various geometry import patterns mentioned above use, in fact, the same generic geometry import pattern, customized for each table in the DB. Consequently, for each table there is a customized data import pattern. This is a generic pattern that sets the existing context – shapes with attached information – from which the following steps can be implemented.

### Basic analysis patterns (BAnalysisPatterns)

These patterns allow querying and mapping the data found in the DB.

*QueryPattern* – The query pattern imports information found in a particular column in a table and, by using a mapping device, called *MappingPattern*, it maps the information onto the site, more or less as in a GIS system. For instance, when one selects *NumberOfFloors*, *this QueryPattern* will select this feature for each building in the *Buildings* table and map it, thereby enabling one to preview the number of floors of each building on site.

### Main composition patterns (MCompositionPatterns)

*ManualGrid* – This pattern is used to open a street in a district area. The pattern is said to be "manual" because it requires action by the designer. The designer is supposed to draw the street axis (a curve or a polyline) and specify the street width. The pattern then generates the street surface and subtracts it from the district surface. However, the pattern allows the generation of multiple streets by assigning multiple curves or polylines to the pattern, hence generating a grid, where each street can be reshaped by manipulating the defining line's control points (Fig. 5c).

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

### *Grid patterns* (GridPatterns)

Grid patterns are structured in a way that they can be used meaningfully by urban designers. In particular, they all share two features: (1) all inputs are meaningful in terms of urban design rationale and (2) all outputs produce the grid street center lines and urban blocks, represented as closed polygons.

*OrthogonalGrid* – This pattern generates an orthogonal grid within a given area. The pattern inputs are: (1) a polygonal area or a set of polygonal areas; (2) the dimensions of the urban blocks in meters ($u \times v$ directions); (3) the street width in the grid also in meters; and (4) the rotation angle to align the grid within the respective polygonal area. Figure 5d shows the application of the *OrthogonalGrid* pattern to several large "islands" subdividing them into smaller rectangular blocks. An additional algorithm, used before the pattern is applied, sets differentiated rotations of the grids according to the longest side of each "island". Another algorithm, applied after the pattern, intersects the grid with the "island" and erases the inner plots, leaving only those along streets (Fig. 5e).

*VoronoiGrid* – The Voronoi grid pattern is based on a Voronoi algorithm. The inputs are: (1) the average block area in square meters and (2) the street width set in meters.

*MergeGrids* – *MergeGrids* intersects two grids. The result can become rather complex. However, this is the method to use every time the designer wants to merge different grids, grid streets with different widths, or to open one street over a grid (like the diagonal in Barcelona). In the latter case, the output of an *OrthogonalGrid* pattern is merged with the output of a *ManualGrid* pattern.

### *Filter-based patterns*

The filter-based patterns work all in the same way. The embedded algorithms apply to a given grid and require placing a point or a set of points as additional inputs, or an additional Boolean condition, which work as selection devices. The filter separates the set of input urban blocks in two different sets: (1) one set with blocks to be treated as standard blocks in the grid and (2) one set with blocks to be treated with different development rules. Depending on the algorithm associated with each specific filter, squares may be opened, exceptional buildings may be generated or simply different rules may be applied to the filtered blocks. The following sample names are reasonably self-explanatory: *SquareInGrid* and *ExceptionalBlock*.

*IgnoreSmallAreas* – This pattern filters grid blocks whose areas are smaller than a certain value specified as the minimum acceptable area for an urban block.

### *Planning patterns* (PlanningPatterns)

The main idea behind planning patterns is to apply planning regulations according to conditions, defined in relation to preexisting elements or to a selected grid or set of urban blocks. The typical condition used in these examples is based on an *AttractorPattern*. The *AttractorPattern* sets an attraction field defined by one or more geometric inputs (e.g. a line or a point). Then, some planning regulations may be distributed according to the attraction field. For instance, construction rules regarding the maximum building height acceptable for blocks in a grid can be set in terms of their distance to the main street set as an attractor. As a result, each block in an urban plan may have a unique set of development rules that stem from the planning conditions defined in such a manner. The regulations are, in fact, generated by the algorithm and can be fed back to the DB as a table of rules and subjected to analytical procedures, as described in Analysis Patterns. Please note that these planning procedures can be applied to a grid being generated from scratch or to an already existing grid. The names of patterns in this category are self-explanatory: *AttractorPattern*, *HeightBasedDistribution*, and *CoverageBasedDistribution*.

### *Analysis patterns* (AnalysisPatterns)

Analysis patterns are of particular interest in terms of planning because they supply real-time information on the properties of the design being generated. In addition, any urban analysis that can be defined using the representation model provided by the DB can be set in the design interface by programming it in the available VPI as an *AnalysisPattern*. The patterns implemented so far calculate the most common indicators that inform decision-making while designing. The most essential of such patterns are density-based patterns. These patterns were based on *Spacematrix* by Berghauser-Pont and Haupt (2010) and they calculate urban indicators, such as building intensity (*FSI* – floor space index), coverage (*GSI* – ground space index), average building height (*L*), spaciousness (*OSR* – open space ratio), and network density (*N*). We started by implementing patterns that calculated these indicators because most plans and regulations use density-based indicators to express goals and constraints.

The use of the indicators mentioned above requires knowledge of the *Spacematrix* theory. The patterns can be applied to calculate density indicators at "district" or "island" aggregation levels. For instance, as a typical workflow, embodying a contextualized design vision, we could consider the following procedural sequence: (1) calculate the average density of the existing blocks or islands (FSI and GSI) and average block size; (2) generate new urban blocks with identical density indicators set as block regulations; (3) define a phased occupation based on three urban elements set as attractors [e.g., two village centers – two-point attractors – and a road connecting both – a curve attractor (Fig. 5f)]. Attractors may influence design features differently by having different weights assigned. Figure 6 shows a design variation for the same site based on changes of attractor weights.

### *Data writing patterns* (DataWritePatterns)

The regulations set by the algorithms mentioned above can be written back into the DB using new patterns to create new tables and columns in a table and populate them with the generated data. The pattern names are self-explanatory: *CreateTable*, *CreateColumn*, and *PopulateColumn*. In this case, the algorithms resort to SQL coding to write the new information in the DB.

### *Street profile patterns* (StreetProfilePatterns)

During a visit to Palmbout office in Rotterdam, after showing the previous patterns to Frits Palmboom, he mentioned that usually, after setting down the plan's layout, he would start sketching the street profile for each street type and then fine tune the street width according to the feedback retrieved from the sketches. In our terms, this means that he was invoking the implementation of patterns to design objects found in the ontological classes SD and SC (Figs. 3 and 4). At that time, this set of patterns had

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

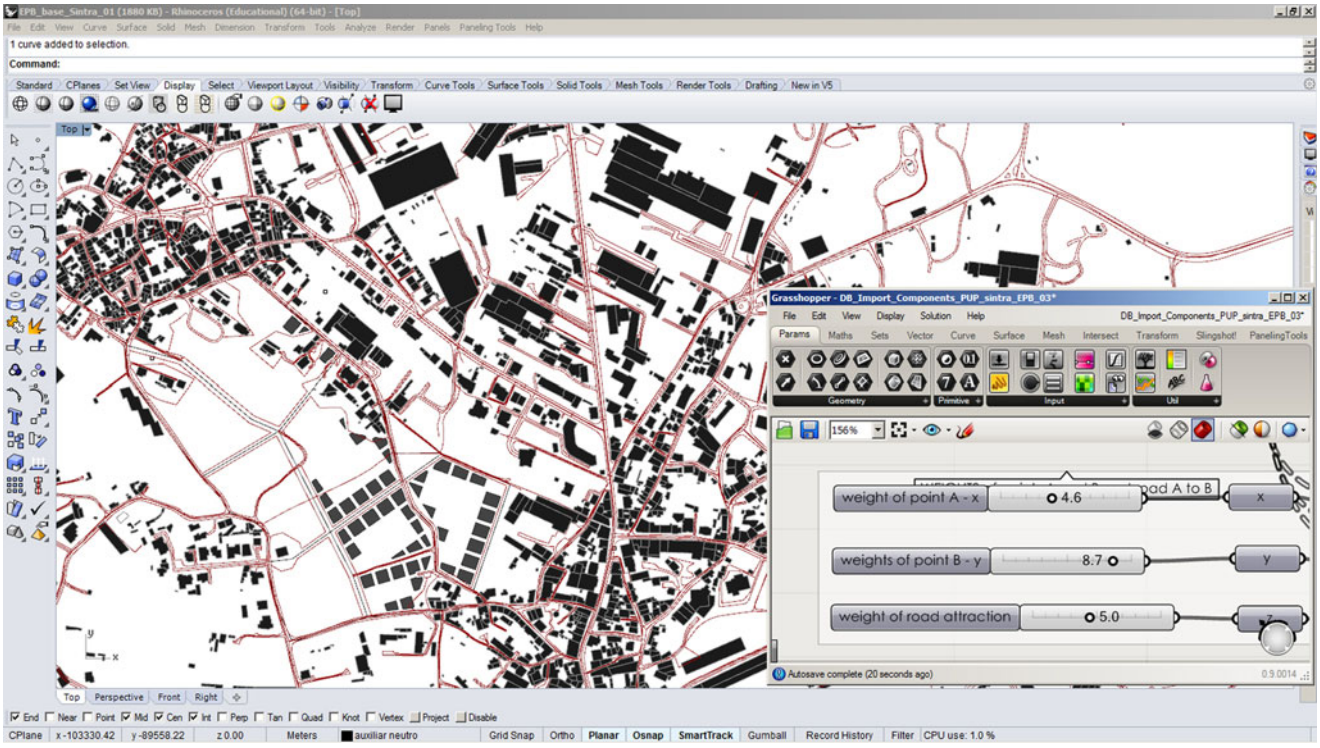



**Fig. 6.** Variations of planned block distribution according to changes in the weight of the attractors.

not been implemented yet, but is now implemented. This feature has been implemented making full use of the ontology's expressed relationships resorting to xml coding and the interface was developed for a regular user without programming knowledge. This topic will be addressed in a specific paper. The ontological structure is shown in Figures 2–4 provides some preliminary information on how the implementation of these patterns was defined.

## Formal definition of generic grammars

After providing basic definitions of generic grammars and the underlying concepts and showing how these grammars can be implemented and applied in the urban design domain, we are in a better position to introduce more elaborate, formal definitions. As mentioned in Section 3, rules have the generic format

predicate → consequent,
or, in a notation more familiar to the domain of production systems, the format
if → then.

Shape grammars, as stated by Gips and Stiny (1980), are production systems where $u$ and $v$ are labeled shapes and $u \rightarrow v$ applies a transformation $t(u)$ to any similarity transformation of $u$ (translation, symmetry, rotation, or scale) found in a design $d$ such that

$$d_1 = [d - t(u)] + t(v) \tag{2}$$

in other words, the design $d_1$ is the result of subtracting a similarity transformation of the shape $u$ from a design $d$ and adding the corresponding transformation of the shape $v$. In a parametric environment, the shape $u$ corresponds to any assignment of values $g(u)$ to the parameters of shape $u$ (Stiny, 1980). Therefore equation [2] becomes

$$d_1 = [d - t(g(u))] + t[g(v)]. \tag{3}$$

Parametric shape grammars may operate on objects found in the ontology shown in Figure 1. They are constrained by the semantic structure embedded in it, producing semantically consistent designs where representations are composed of shapes and symbols representing parts of the urban environment and other descriptions that convey semantic consistency to the overall design. Typically, urban design patterns are defined by shape grammars operating within a single class of the ontology. These are algebras of the format $\gamma_i = \{S_i, L_i, R, I_i\}$, where $S_i$ and $L_i$ are shapes and labels from an object class $C_i$, $I_i$ is an initial labeled shape in the same class, $R$ is a set of rules $R$ of the form $u \rightarrow v$ that apply to objects $u$ and $v$ from class $C_i$. Designs produced by such patterns are kept within the same class of the ontology ($\gamma_i \rightarrow \gamma_i$). An urban design pattern is triggered when its initial shape is placed in the design as a result of the instantiation of previous patterns by the corresponding shape grammar. These are the most common urban design patterns in an Urban Grammar $\Gamma'$ (Fig. 7). Patterns initiating the design generation process are called initial patterns. In urban design, initial patterns have as their initial shapes a preexisting object $E$ in the existing urban context, set as a reference object $R_{ef}$. Rules of these kind transform $E$ into some object $u$ where $u$ is a labeled shape belonging to a class $C_i$.

$$E \rightarrow u,\ u\ \varepsilon\ C_i = \{S_i, L_i\}. \tag{4}$$

These rules allow the start of design generation from a particular object class $C_i(E \rightarrow \gamma_i)$; in other words, they initialize a specific

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

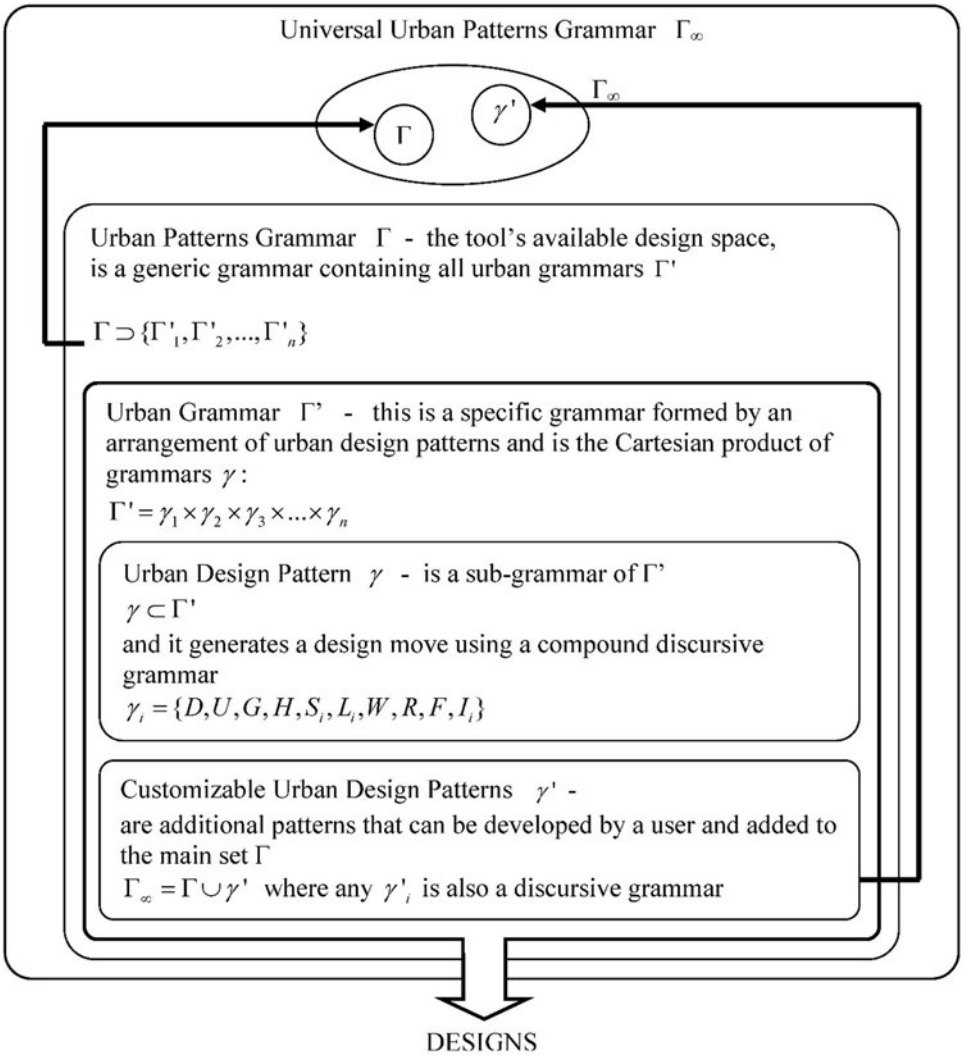



**Fig. 7.** The Universal Urban Patterns Grammar and all its sub-grammars.

type of urban objects in the design. It also means that the patterns that were previously referred to, those operating within the same object class, can now be applied. In addition to patterns of the form (4), some patterns can transform objects from one class into objects of another class. For instance, urban blocks (BL) can be transformed into buildings (BD). There are two types of these patterns. The difference between them lies in whether the rules transform objects from an upper level of the ontology (or level of abstraction) into those of a lower or vice versa, that is, whether they have the format $(\gamma_i \rightarrow \gamma_j)$ or $(\gamma_j \rightarrow \gamma_i)$. The former defines the semantic relation *u has v*; while the latter defines the relation *v is part of u*. In the former, rules assume the format:

$$u \rightarrow v,\ u\ \varepsilon\ C_i \wedge v\ \varepsilon\ C_j \mid pr(C_i, C_j) \tag{5}$$

where *pr* stands for the primary relations given in Figure 1 constraining $C_i$ to be of a higher level of abstraction than $C_j$, the semantic relation is *u* has *v* and the algebra $\gamma_i$ permits one to apply a new algebra $\gamma_j$ allowing for further detailing of the design. Conversely, the relation *v is part of u* produces rules of type:

$$u \rightarrow v,\ u\ \varepsilon\ C_j \wedge v\ \varepsilon\ C_i \mid pr(C_i, C_j) \tag{6}$$

where an algebra $\gamma_j$ allows to further develop parts of the design pertaining to an upper level of abstraction by generating new objects from an already opened algebra $\gamma_i$. These rules allow for fine-tuning the design enabling what in "design thinking" theory could be called corrective feedback loops. The several classes in the ontology help keeping the semantic structure clear but they also provide a clear means to separate design phases.

The idea behind the "Pattern Language" concept (Alexander et al., 1977) was to define an algorithmic approach to architectural and urban design by providing common design instructions to solve well-known design problems. These instructions define verbal descriptions of the conditions required to apply a solution, also described verbally. In this paper, patterns describe typical urban design operations both verbally and algorithmically in the form of a discursive grammar (Duarte, 2005) by specifying the rules (*R*) that generate such design operations. In this case, patterns specify the objects they operate on, the classes they belong to, the class of objects they can generate and sometimes some constraints on shape attributes, thereby defining predicate conditions for pattern application and providing semantic guidance to rule application. Furthermore, patterns also state that their consequents belong to specific classes or are constrained to specific objects, attributes, or parameter ranges. The whole

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

set of urban design patterns forms a generic grammar for urban design which can be applied in, possibly infinite, many different contexts.

In more detail, the complete set of patterns defines a generic grammar for urban design – the Urban Patterns Grammar Γ. Considering a hypothetical system where additional design patterns $\gamma'$ may be customized by the grammar user, the union of all grammars in the Urban Patterns Grammar Γ with all customizable grammars $\gamma'$ defines a theoretically infinite set of grammars that can generate any kind of urban design – the Universal Urban Patterns Grammar, $\Gamma_\infty$. The Urban Design Patterns, called UIP in CItyMaker (Beirão et al., 2012) replicate with a fair degree of flexibility a typical urban design operation. They are given by a discursive grammar of the format

$$\gamma_i = \{G, D, U, H, S_i, L_i, W, R, F, I_i\}. \quad (7)$$

The discursive grammar format addresses the complexity that semantic contextualization brings to urban design problems. $U$ is a set of urban design regulations, that is, of constraints inherited from upper-level plans or from local or national legislation. $G$ is a set of symbolic descriptions used to describe contexts, design goals, and designs. $D$ is a set of description rules either used to generate the design goals from a context or to describe the designs being generated. Goals are the result of an urban program formulation procedure using tools and methods described in (Montenegro et al., 2011). Designs are obtained starting from an initial shape $I_i$, and applying rules $R$ on objects taken from a class $C_i$ in the urban ontology composed of shapes and labels $\{S_i, L_i\}$ where $S_i$ and $L_i$ may sometimes be an empty shape and an empty label, depending on the class they represent. The generation of designs is guided toward the defined goals by resorting to a set of heuristics $H$ based on the sequence of decisions defined at the beginning of Section 4. $W$ is a set of weights, symbolic, or visual markers used to qualify formal descriptions, in order to assign meaning to shapes or to provide control devices over the generations of designs. $F$ is a set of functions used to impose constraints on the way both description and shape rules can be applied, thereby providing additional control devices over design generation. For instance, they may relate regulations in $U$ to symbolic descriptions in $D$ or shape descriptions in $S$.

## Implementation and methodological issues

Shape grammar implementations are usually confronted with difficulty in finding interpreters advanced enough to provide a friendly interactive environment for designing and for developing complex designs involving complex semantic conditions. Furthermore, any shape grammar implementation requires a deep discussion of the role of emergence within the design environment, and especially on the semantic control of emergent shapes in the design. Contrarily, parametric design interfaces are well developed and provide very interactive platforms, some of them quite in tune with what has been established through "design thinking" theory as responsive practice – continuous cyclical responses to trial design moves. Schön's see-move-see cycle (1983) shows with particular accuracy the importance of visual *stimuli* to design decision. Considering that parametric design environments generally provide the required *stimuli* that most designers request from a design platform, we decided to adapt the generic shape grammar concept to a generic parametric pattern concept as described in Section 4. The obtained structure

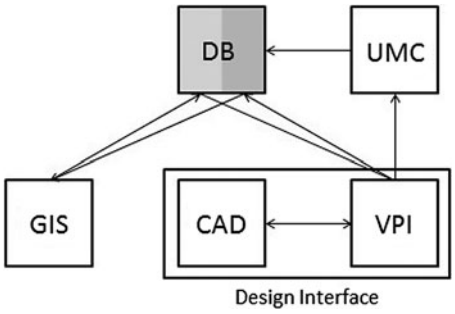



**Fig. 8.** Diagram of the basic structure of a City Information Modeling (CIM) platform: DB, date base; GIS, Geographic information system; CAD, computer-aided design; VPI, visual programming interface, UMC, measures and calculations unit.

forms what we could call a City Information Modeling (CIM) platform because it comprises the access to geographic information, spatial analysis tools, and parametric design tools.

Following an analogy with BIM (Building Information Modeling), CIM integrates into a single platform, urban design, and assessment tools. Broadly, a CIM platform is in accordance with the structure diagrammed in Figure 8 and described in the following. DB is a geographical database that can be read by a GIS or by a design interface composed of CAD software and a VPI that reads data from and writes data to DB. The two shaded areas represent the distinction between existing contextual data (light gray) and new design data generated by the design interface (dark gray). The CAD + VPI platform is structured as a programmable parametric design interface that uses a generic grammar for the generation of urban designs. The generic grammar is defined as described in the previous sections and is composed of several "urban design patterns" implemented as parametric design patterns in the VPI. Different combinations of "urban design patterns" with different input data produce different urban plans which can then be subjected to an evaluation in context using the UMC module. The UMC module, in association with the VPI, calculates the urban indicators that correspond to the urban models being generated. This process occurs in real time as indicators are immediately updated whenever changes to the design are introduced by changes in the input.

The implementation described above calculates density indicators of plans following the premises defined by Berghauser-Pont and Haupt (2010). The parametric platform described above allows the user to generate urban plans by composing urban patterns and by choosing the values of parameters associated with the patterns. It also provides the user with immediate feedback expressed in terms of urban indicators, thereby enabling a control of the outcome. The urban plans produced with this platform are flexible as the platform can be used to regenerate alternative solutions by manipulating patterns and parameters.

## Discussion and conclusion

This paper proposes, describes and defines the concept of generic grammar and illustrates its implementation with the development of a generic grammar for urban design. The goal was not to automate the design process but to provide a platform to support and facilitate the designers' work. However, the platform has built-in features that speed up the process of generating designs and assessing their impact, thereby allowing the designer to make more informed design decisions, which might revert in design quality.

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

A generic grammar includes an ontology describing the domain to which the generic grammar is applied and a set of grammar based patterns. It is argued here that the generic grammar framework as defined in Section 5 can be applied to other design domains following similar definitions. For such purposes, domain-specific ontologies need to be created as well as specific patterns.

The proposed generic urban grammar was converted into a parametric design model and then implemented using Rhinoceros and Grasshopper as the core implementation environments. This conversion was necessary to facilitate implementation as there are no robust shape grammar interpreters currently available and our concern was to test the validity of the proposed generic framework. Given this conversion of the grammar into a parametric design model, one may ask why not proceed directly to the parametric model. However, our experience shows that developing the grammar is a necessary first step. The grammar paradigm provides a very useful knowledge-representation model that combines both visual and symbolic information. Moreover, it allows one to proceed incrementally by encoding partial knowledge into one or more sets of rules, that is, into small grammars. It was this affordance of shape grammars that triggered the idea of encoding patterns as small grammars and, therefore, led to the concept of generic grammars. In summary, shape grammars provide the necessary technical apparatus – shapes, descriptions, labels, parameters, and so on, – to organize in an incremental fashion the various types of information associated with design knowledge.

When applied to the urban design domain, the generic grammar framework provides a CIM environment. The main characteristic of such platform is joining analysis, synthesis, and evaluation in a unique design environment. As such, the CIM environment embedded in the implemented computer platform is simultaneously a design platform that supports the generation of design scenarios, a simulation platform that permits to assess such scenarios, and a decision support platform that allows decision makers to discuss and evaluate the consequences of proposals. For this reason, it can also constitute an instrument of dialogue between the various stakeholders. It is curious to note that the parametric implementation favors its use as a simulation tool because some, possibly judiciously chosen indicators are available to show the impacts of design changes, hence keeping a register of alternative designs and the results of their assessment. Nevertheless, it should be stressed that the platform does not replace the role of interpretation from the specialized urban designer and team. It only provides information in real time while testing design scenarios facilitating the understanding of the relation between design solutions and their performance attributes.

In summary, the concept of generic grammars seems to provide a useful paradigm for the development of the next generation of CAD platforms. Future work will be concerned with extending the urban design platform and with continuing work on the application of the concept to other design domains.

## References


**Alexander C, Ishikawa S and Silverstein M** (1977) *A Pattern Language*. Oxford: Oxford University Press.

**Beirão JN** (2012) City Maker: designing grammars for urban design, PhD Dissertation, Delft University of Technology, The Netherlands.

**Beirão JN and Duarte JP** (2009) Urban design with patterns and shape rules. In EH Stolk and Brömmelstroet M (eds). *Model Town: Using Urban Simulation in New Town Planning*. Almere, The Netherlands: New Town Institute, pp. 148–165.

**Beirão JN, Duarte JP and Stouffs R** (2010) Creating generic grammars from specific grammars: towards flexible urban design. *Nexus Network Journal* **13**(1), 73–111.

**Beirão JN, Duarte JP, Stouffs R and Bekkering H** (2012) Designing with urban induction patterns: a methodological approach. *Environment and Planning B* **39**, 665–682.

**Benrós D, Duarte JP and Hanna S** (2014) The inference of generic housing rules: a methodology to explain and recreate Palladian Villas, Prairie Houses and Malagueira Houses. In Gero J and Hanna S (eds). *Design Computing and Cognition DCC'14*. The Netherlands: Springer, pp. 439–458.

**Berghauser-Pont B and Haupt P** (2010) *Spacematrix: Space, Density and Urban Form*. Rotterdam: NAI Publishers.

**Caldas L and Norford L** (2002) A design optimization tool based on a genetic algorithm. *Automation in Construction* **11**(2), 173–184.

**Castro e Costa E and Duarte JP** (2013) Mass customization of ceramic tableware through digital technology. In Bártolo MH, Bártolo P, Alves N, Mateus N, Almeida H, Lemos A, Craveiro F, Reis C, Reis I, Durão L, Ferreira T, Duarte JP, Roseta F, Castro e Costa E, Quaresma F and Neves JP (eds). *Green Design, Materials, and Manufacturing Processes*. The Netherlands: CRC Press/Balkema, pp. 467–471.

**Coelho C, Costa J, Leite J, Silva J, Trindade L, Pereira P, Proença S, Fernandes S and Monteys X** (2013) *Cadernos de Morfologia Urbana: Os Elementos Urbanos*. Lisboa: ed. Argumentum.

**Duarte JP** (2005) A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira. *Automation in Construction*, **14**(2), 265–275.

**Duarte JP and Beirão JN** (2012) Towards a methodology for flexible urban design: designing with urban patterns and shape grammars. *Environment and Planning B* **38**(5), 879–902.

**Duarte JP, Beirão JN, Montenegro N and Gil J** (2012) City induction: formulating, generating, and evaluating urban plans. In Müller Arisona S, Wonka P, Aschwanden G, Halatsch J (eds). *Digital Urban Modeling and Simulation. Communications in Computer and Information Science (CCIS)*, vol. **242**. Berlin, Heidelberg: Springer, pp. 79–104.

**Fleisher A** (1992) Grammatical architecture? *Environment and Planning B* **19**(2), 221–226.

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

**Garcia S and Barros M** (2015) A grammar-based system for chair design: from generic to specific shape grammars. In Martens B, Wurzer G, Grasl T, Lorenz WE and Schaffranek R (eds). *Real Time - Proceedings of the 33rd ECAADe Conference - Volume 1, Vienna University of Technology, Vienna, Austria, 16–18 September 2015*, pp. 427–436. CUMINCAD. http://papers.cumincad.org/cgi-bin/works/paper/ecaade2015_233

**Gips J** (1999) Computer implementation of shape grammars. Proc. Workshop on Shape Computation, Cambridge, MA, June.

**Gips J and Stiny G** (1980) Production systems and grammars: a uniform characterization. *Environment and Planning B* **7**(4), 399–408.

**Gruber T** (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**(2), 199–220.

**Hillier B and Hanson J** (1984) *The Social Logic of Space*. Cambridge: Cambridge University Press.

**Hillier W, Hanson J and Peponis J** (1987) Syntactic analysis of settlements. *Architecture and Behaviour* **3**(3), 217–231.

**Knight T** (1992) Designing with grammars. In Schmitt G (ed.). *CAAD Futures '91, Computer Aided Architectural Design Futures – Education, Research, Applications*. Braunschweig and Wiesnaden: Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, pp. 19–34.

**Koning H and Eizenberg J** (1981) The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B* **8**, 295–323.

**Lehnerer A** (2009) *Grand Urban Rules*. Rotterdam: OIO.

**Marshall S** (2005) *Streets & Patterns*. London and New York: Spon Press.

**Marshall S** (2011) *Urban Coding and Planning. Introduction*. Oxfordshire, UK: Routledge.

**Mendes L, Beirão J, Duarte J and Celani G** (2013) A Bottom-Up Social Housing System Described with Shape Grammars. *Proc. eCAADe 2013 Conf.*, Delft, The Netherlands, September 18–20.


https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press

**Montenegro N, Gomes J, Urbano P and Duarte J** (2011) 4CitySemantics: GIS-Semantic Tool for Urban Intervention Areas. Proc. 7VCT Conf., October 11–13.

**Muratori S** (1967) *Civiltà e territorio*. Roma: Centro studi di storia urbanistica.

**Post E** (1943) Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics* **65**(2), 197–215.

**Schön D** (1983) *The Reflective Practitioner: How Designers Think in Action*. New York: Basic Books.

**Shea K and Cagan J** (1997) Innovative dome design: applying geodesic patterns with shape annealing. *Artificial Intelligence for Engineering, Design and Manufacturing*, **11**(5), 379–394.

**Stiny G** (1980) Introduction to shape and shape grammars. *Environment and Planning B* **7**(3), 343–351.

**Stiny G** (1981) A note on the description of designs. *Environment and Planning B*, **8**(3), 257–267.

**Stiny G and Gips J** (1972) Shape grammars and the generative specification of painting and sculpture. *Proc. Int. Conf. Information Processing 71*. Amsterdam: NorthHolland.

**Stiny G and Mitchell W** (1978) The Palladian grammar. *Environment and Planning B* **5**(1), 5–18.

**José Beirão** obtained a professional degree in Architecture from the Technical University of Lisbon, Portugal, in 1989, and worked at Gonçalo Byrne and Falcão de Campos offices, thereafter. In 1998, he founded the architecture firm Bquadrado architects together with Miguel Braz. He obtained a Master degree in Urban Design from ISCTE-IUL, the University Institute of Lisbon, in 2005 and completed his PhD in Urbanism at TU Delft, The Netherlands, in 2012. The theme of his dissertation is the development of design patterns for the establishment of computational platforms for urban design. His current research interests focus on the use of parametric systems and geographic databases to investigate the following concepts: (1) measuring parameters of urbanity and morphological studies, (2) development of urban design evolutionary systems, (3) customizable systems for social housing including actions at the urban level, and (4) developing strategies for the Portuguese dispersed territories.

**José P. Duarte** holds a professional degree in Architecture from the Technical University of Lisbon, Portugal, a Master's degree in Design Methods, and a PhD in Design and Computation from MIT, USA. Currently, he is Chair in Design Innovation and Director of the Stuckeman Center for Design Computing at Penn State University, USA. He was Dean and Professor at the Faculty of Architecture, University of Lisbon, and President of eCAADe – education and research in computer-aided architectural design in Europe. The main focus of his research is the use of new technologies as conceptual tools in architectural, urban, and product design. He was PI of the City Induction research project funded by the Portuguese Foundation for Science and Technology (FCT).

https://doi.org/10.1017/S0890060417000452 Published online by Cambridge University Press