Ontologies and Shape Grammars: Communication between Knowledge-Based and Generative Systems

Francois Grobler, Ajla Aksamija and Hyunjoo Kim
US Army Corps of Engineers, ERDC Construction Engineering Research Laboratory, USA

Ramesh Krishnamurti, Kui Yue and Casey Hickerson
Carnegie Mellon University, USA

This paper discusses information flow between knowledge-based models and shape grammars for generation of building designs, explaining the interaction, system and implementation. The benefit for using the interactive system is that the complementary properties of the two schemes are used to strengthen the overall process. Shape grammar contains rules about the geometric organization, while knowledge-based model supports the contextual information.

Introduction

The nature of architectural design poses immense challenges for computing and information processing in automated or semi-automated systems. Architectural design knowledge, thinking and process are crucial components in the overall course of creating buildings; yet, computational representations of these components have been the central issue [1], [2], [3], [4]. In particular, types of information that architects seek vary depending on the nature of the problem, and the method in which information is sought is often ambiguous. For example, demographic studies are important to understand the social context of a particular site, building codes are crucial to understand the laws and regulations, and weather data is important to understand the environmental aspects. This explicit knowledge is stored in databases, is easily accessible by the designer and can be represented and

J.S. Gero and A.K. Goel (eds.), *Design Computing and Cognition '08,*

analyzed computationally. The source of inspiration and implicit design knowledge are much more difficult to express. Moreover, computational analyses of sources of ideas and design require thorough understanding and comprehension of intentions, as well as contextual aspects. Designer intention and thought process are indeterminate from the system point of view. In this work, knowledge-based and generative systems are combined to construct a method for analysis of building types, in particular layout generation depending on certain parameters, such as building location and dimensions. Shape grammar contains rules about geometric manipulation and transformation, while knowledge-based model contains information about spatial use, organization, elements, and contextual information. Buildings are analyzed and layouts are generated through communication and interaction between these two systems. A case study of Baltimore rowhouses is used to illustrate the process.

Aims and Significance

Ontology contains information about building designs, location, use, orientation, and size, but does not give form to buildings with geometric meaning. It is knowledge representation about a subject, and describes individuals as basic objects, classes as collections or types of objects, properties and characteristics, and relations between objects. In this research, ontology is used to capture knowledge relating to architectural design principles, building anatomy, structure and systems. There are certain similarities between shape grammar and knowledge-based model, mainly that both contain design rules, however, the nature of the rule varies.

Shape grammars are computational rules for generation of geometric shapes, and there are two types—analytic and original. Analytical grammars are developed to describe and analyze historical styles or designs by specific architects [5], [6], [7], [8]. Analytical grammars use sets of existing designs, the corpus, to develop the language, and to infer the rules. Grammars are tested by using the rules to both generate designs in the corpus, as well as new designs. Original grammars are based on generalized rules and are intended to create instances of original styles of designs. These types of grammars have not been widely addressed as analytical, owing to the difficulty of "translation of abstract, experimental form into architectural designs that fit particular design contexts or programmes" [9].

The combinatory nature of the design rules captured by shape grammar and ontology offers the possibility that design knowledge can be explicitly represented, maintained and processed. In this respect, Baker and Fenves [10] remark:

In an engineering scenario, analysis and design are the similar processes manipulating constant knowledge. When engineers use knowledge to create new products the process is called design. The process of creating new objects will be termed generation. The checking process of objects will be called critiquing, otherwise known as analysis. Therefore, if engineering knowledge can be captured (e.g., how engineered objects fit together as well as the function of each object) and represented using a grammar, both design and analysis can be performed.

A parallel can be drawn between engineers and products, and architects and buildings. Architectural design knowledge is captured in ontology, and processed by shape grammar, thus allowing for generation and analysis. In this sense, any discrepancy between analytical and original shape grammars become diminished, as general rules can be customizable depending on context, building size, style or function.

Application

In a traditional analytical shape grammar, as found in the literature, we often come across rule descriptions of the form: "If the back or sides are wide enough, rule 2 can be used …," which are inherently counter-computable. For shape rules to be "computation-friendly," rules need to be quantitatively specified so that they translate easily into pieces of "code," and that there is enough precision in the specification to disallow generation of ill-dimensioned configurations. A computation-friendly shape grammar interpreter would benefit from assistance of the ontology. In our efforts to quantify shape rules, originally specified in the traditional way, we frequently found that the only way to distinguish certain rules is employ threshold values statistically derived from the building sample, for example, for the area of a kitchen. The ontology can provide this dynamically as new building samples are added.

In the context of this paper, it is important to note that shape grammars are primarily used as a knowledge base for the building geometric forms, as well as a vehicle for geometric derivation for layout generation. Generating novel designs is not a concern of this particular research.

Methodology: process and communication

The starting point of the interaction focuses on requirements, including building location, dimensions, and functional type. A lightweight building

information model (BIM) is created based on the requirements, and it contains information about the building shell, general building anatomy, and components. This data is the initial input for the shape grammar rules, but additional information about the building, such as site context, environment, and cultural effects, needs to be incorporated in the process. The means of providing that information to the shape grammar rules is through the ontology of architectural design drivers. This model includes specific information according to the building type, location, culture, environment, structural system, and context, and about design factors that directly influence building layout. Once that information is received, the shape grammar system selects the rules and configures the spatial organization. During the application of shape rules, the shape grammar system may query the ontology system for certain information, in particular, statistical data and facts. The queried data will be used to decide which rule to apply among the candidates for the next step. Such queries are currently designed in a way so that no human intervention is necessary although this is not true in general. The end result is a generated layout, outlining spatial organization. The information populates the parametric BIM, and can be visualized in three dimensions. Figure 1 presents the overall process and the interaction.

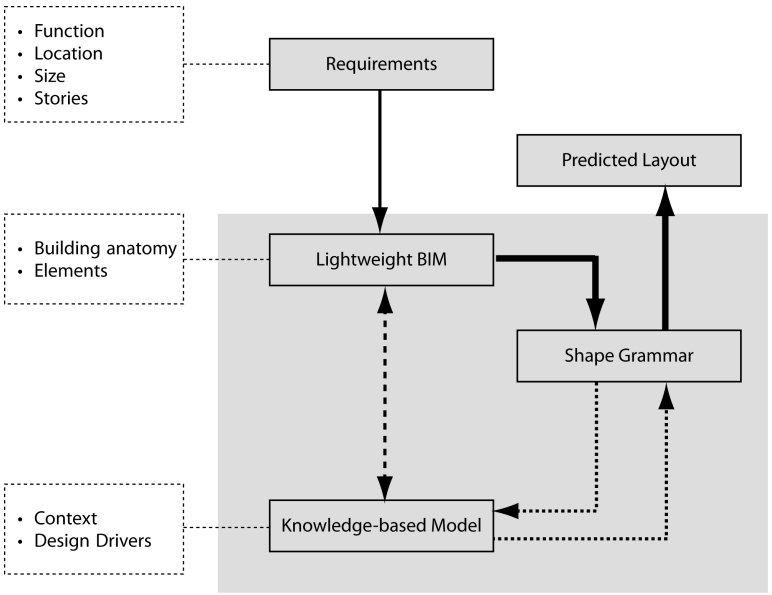

Fig. 1. The overall process and interaction between shape grammar and knowledge-based model

Lightweight building information model

The purpose of developing a lightweight BIM is to build a neutral, semantic, slimmed down representation of a building, objects in that building and relationships among them and thus to capture common building elements and anatomy [11]. Building anatomy is subdivided into gross building elements such as external features and general descriptions where each gross building element is further divided into detailed elements such as roof, walls, floors, foundation, doors and windows. This research utilized an ontological structure to represent a common behavior of a building and provide an underlying structure of objects and relationships of a building where different relations such as "bounding_Walls," "connects_Rooms," "is_Made_Of" and so on are described between elements. Figure 2 shows an ontological instance of a building with different properties and relations to represent the lightweight building elements of a Baltimore rowhouse.

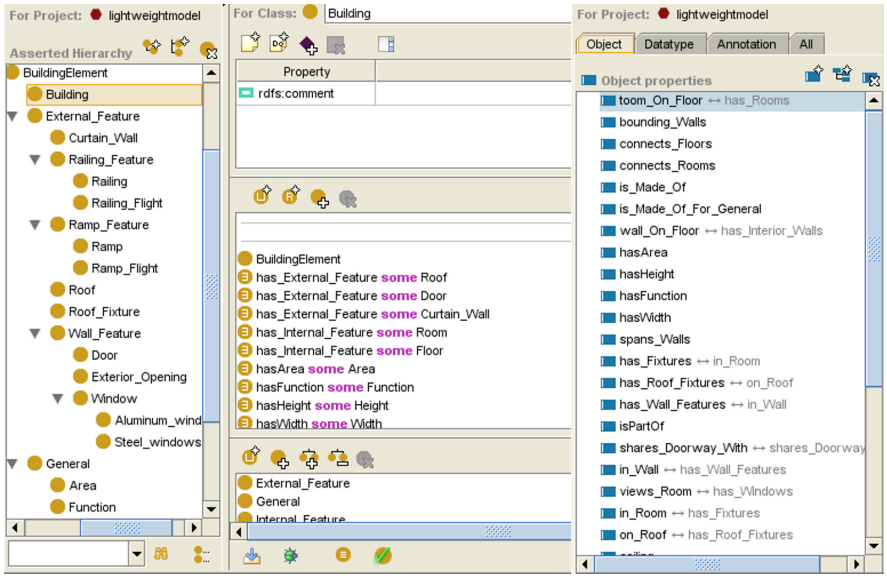

Fig. 2. Light-weight BIM contents

Knowledge-based model of architectural design drivers

The purpose of the architectural design driver ontology is to capture knowledge about the design of particular building types. The features of a building that relate to the design include size and dimensions, surrounding area, façade treatment, circulation and movement patterns, form, structure, materials, etc. The ontology is used to describe the relationships between these elements, as well as the social, cultural, and environmental factors.

The main classes include environmental, social, cultural and physical properties—the rest of the ontology follows a hierarchical model describing the main concepts. Environment class defines context, infrastructure, and site. Context includes cultural aspects, history, economy, social aspects (such as users and activities), and style. The ontology contains logical descriptions of the relationships between these aspects, thus constructing representations of these implicit drivers. Site is defined through human-generated and natural classes, such as density, climate, and topography. Function class defines the use of building, whether it is residential, commercial, industrial, or institutional. Shape class defines the two and three-dimensional types of forms and geometry. System class defines the mechanical and structural systems within a building, as well as materials. Mechanical systems include electrical, plumbing, lighting, and heating, ventilation and air-conditioning (HVAC) system. Structural systems include components, such as beams, columns, and trusses, as well as different types, such as wood, masonry, concrete, steel and composite systems. The necessity for defining mechanical system is not directly related to the queries by shape grammar, but rather for the textual representation of additional information about particular designs.

The overall process needs to support buildings located in different geographical areas. Specific building types, such as hotel, theater, hospital, office, house, high-rise, etc., are defined using the properties and relationships to the declared classes, such as required spaces, elements, and types of construction [12]. Design rules depend on physical systems, location, environment, culture and building function, and as such are used to construct the knowledge-based model. Further, logical restrictions are used to express the dependencies between building types and the activities, components, budget, circulation, location, etc., as presented in Figure 3 for manufacturing plant as a specific type.

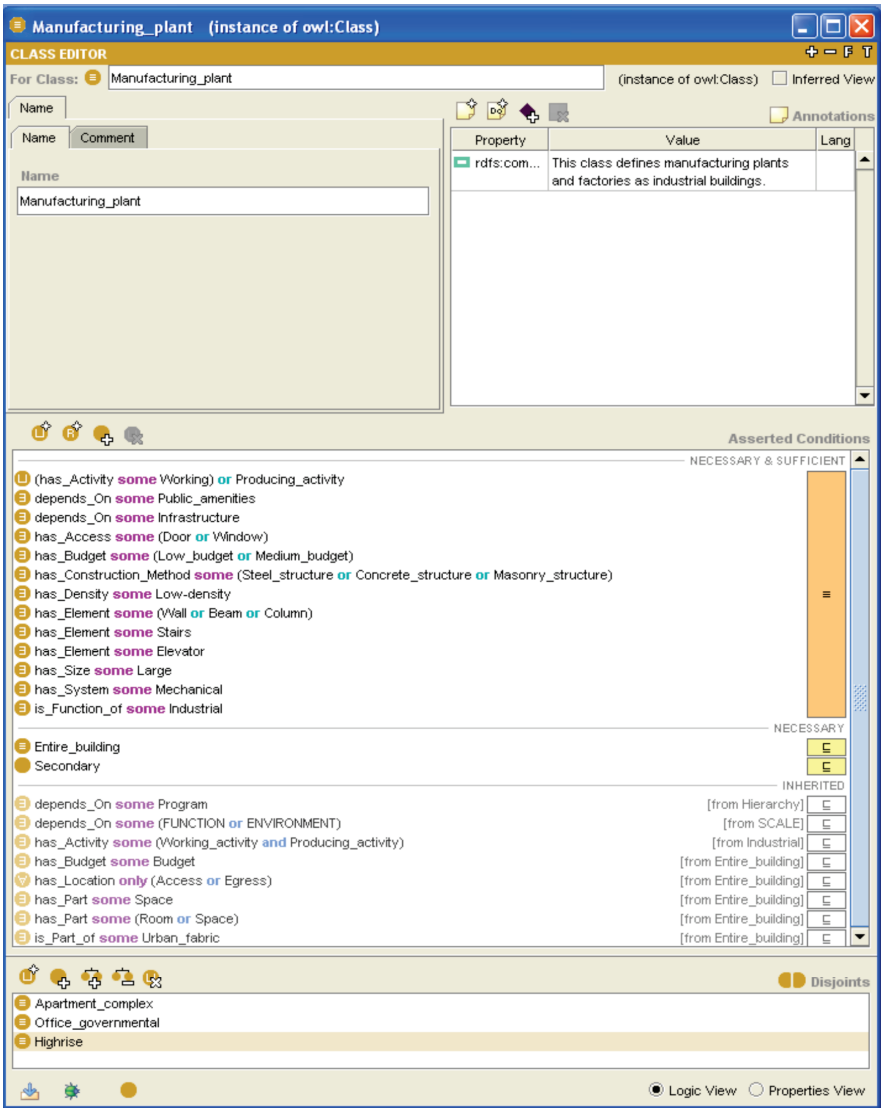

Fig. 3. Logical descriptions for manufacturing plant

Shape grammar and layout generation

In general, given present day technology, it is difficult for a machine to generate building interior layouts from a set of observable exterior features. However, by using knowledge of building styles this task (or a sub-

set) could be made significantly more tractable. Many buildings follow a pattern book; that is, they vary according to well-defined configurational patterns as well as certain established sets of regulations and dimensions. Shape grammars offer the facility of capturing the spatial and topological aspects of building styles within a rigorous formalism. As such, grammars can be used to generate building designs.

The challenge is to use a base of general design knowledge about buildings in a given style coupled with limited specific knowledge about a building with the purpose of generating its interior layout. Formally, any such algorithm for layout generation requires three main types of information:

- Building footprint
- Set of exterior features: windows, surrounding buildings, chimneys, etc.
- Shape grammar

Note that by the assumption of the availability of a shape grammar, the underlying buildings implicitly have a clear and rigid spatial organization; this greatly narrows the scope of buildings under investigation. Moreover, in principle, when applied exhaustively, shape grammars generate, as a tree, the entire layout space of a style. By leveraging this fact, we can specify an approach, which begins with an initial layout estimate obtained from employing building feature constraints on the feature input. From this estimation, further spatial and topological constraints are extracted. These constraints are then used to prune the layout tree. The layouts that remain correspond to the desired generations.

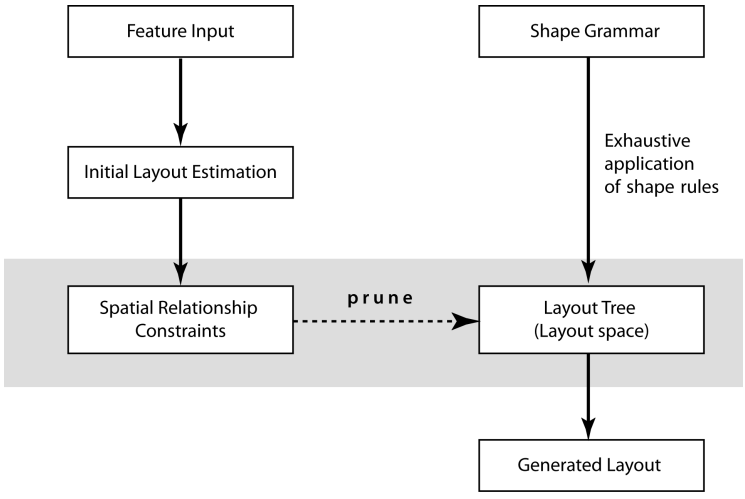

Fig. 4. Approach for layout generation

Results: case study

Queries for design rules

The Baltimore rowhouse [13] is discussed as a case study to demonstrate the process and communication between knowledge-based and generative systems. As the shape grammar rules start to execute, the system poses queries to ontology, such as the required or optional features, required spaces, dimensions, and orientation. This information is received, processed, and according to the responses the subsequent rules are implemented.

- The starting point for the interaction between shape grammar and the ontology is the list of questions that shape grammars poses for a specific building type. The questions for Baltimore rowhouse inquire about the building orientation, surrounding context, spaces, dimensions, and construction method:
- Which direction is the front?
- Which sides of the building face streets?
- Which direction is north?
- What kinds of exterior features are common to the type?
  — *Door, window, chimney, porch, dormer*
- What kinds of interior features are common to the type?
  — *Fireplace, stair*
- What kinds of spaces are common to the type?
  — *Hallway, parlor, kitchen, dining-room, air-lock*
- What kinds of wall assemblies are common to the type?
- Of the various features and spaces common to a type, which are required?
- How do interior spaces relate to building orientation?
  — *In the Queen Anne style and Baltimore rowhouse types, the parlor always faces the front side of the building.*
- How do exterior features relate to interior spaces?
  — *A front door is always on the front side of the building, though in the Baltimore rowhouse style, a front door does not always enter a hallway.*
- How do interior features relate to exterior features?
  — *The interior fireplace is offset from the chimney on the exterior.*
- What are the minimum, maximum, and average/expected dimensions for features, spaces, and wall assemblies?

— *For fireplace (depth), staircase (slope), staircase (run), parlor (width), parlor (depth), hallway (width), hallway (depth), interior-wall (thickness) and so on.*
- Do features align with stories?
- Is a group of features symmetric? If so, what is the symmetric axis?

Interaction

Baltimore rowhouses are typically quite narrow, two stories high, and facing north-south or south-west. Living rooms typically face front, and are directly accessible from the street or narrow hallway if there are two bays, and kitchen is located in the back [13]. Wood stairs are often a single run, and are oriented along one firewall. The fire walls are primarily constructed of brick, with wood framed structure for interior partitions, floors and roofs. Roofs are typically with nominal slope.

The information presented above is captured in the architectural design drivers ontology through different methods. For example, general building class contains elements, in which case an instantiation of Baltimore rowhouse presents actual elements of this particular building type. Fire wall is one of the key elements of a rowhouse, and the spatial organization is always linear and dependent on this key element. Similarly, building spaces belonging to the Baltimore rowhouse are captured. The spatial organization, general rules and typical sizes are also captured as instances, where statements such as "living room faces front" are constructed from the elements of the ontology. Minimum, maximum and average dimensions are presented for all spaces. Spatial organization is presented relative to the external and internal features. Ontology also contains information about selected existing buildings, such as footprint, material use, statistics of spatial use and organization.

Interaction between shape grammar and ontology is accomplished through XML communication by web services. Thetus Publisher[1] contains the ontology discussed in this paper. The tasks for Thetus Publisher include collecting, storing, structuring, changing and searching knowledge bases. Specific queries can be saved and stored in Thetus, as well as reused and updated as the knowledge is discovered. In the layout generation process, the queries are performed by shape grammar, where specific questions are asked from the ontology. The questions are directed to Thetus, which gathers the necessary information and sends to shape grammar rules.

[1] A knowledge modeling and discovery environment developed by Thetus, Inc. (http://www.thetus.com/)

Figure 5 presents architectural design drivers ontology for Baltimore rowhouse and constructed search for relationship between exterior and interior features. Once the information is received, the generation process initializes. The generated layouts are sent back to Thetus Publisher as XML-BIMs, where they are stored.

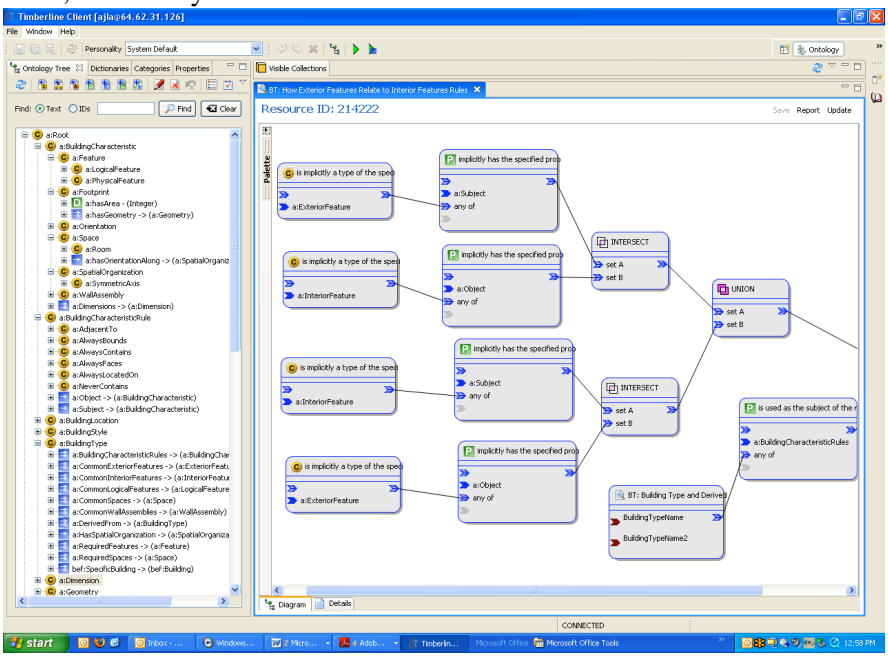

Fig. 5. Ontology capturing design knowledge for Baltimore rowhouse and constructed query for relationship rules between exterior and interior features

There are several technical issues associated with the implementation of communication through web services. Thetus Publisher is the master program of the entire building characterization system, and the generative system, named PILOT (Proposing Interior Layout Over building Types), is a sub-system. The basic model of web services is query-and-response; that is, one side starts a HTTP query of the form http://64.xx.xx.xxx/BuildingTypeServlet/?action=runSearch&search=commonSpacesForABuildingType&buildingType=BaltimoreRowHouse, and the other side writes an XML response back. Due to the limitation of the query-and-response model, certain communications have to be realized by multiple query-and-responses.

Table 1 XML communication protocol

Function	Publisher's initial request to PILOT with building feature inputs.
Query	PILOT?action=generationRequest&buildingType=BaltimoreRowhouses PILOT will initialize and start a generation thread. After dispatching the thread, PILOT will respond immediately, without waiting for the generation thread to terminate
Cases	**Xml response**
Succeed in dispatching a thread	<response status="success"> <msg>…</msg> <generationId>12</generationId> </response>
Fail in dispatching a thread. Return -1 generation ID.	<response status="fail"> <msg>…</msg> <generationId>-1</generationId> </response>
Query	Publisher?action=featureInputRequest&generationId=123 PILOT queries Publisher for XML feature inputs.
Case	**Xml response**
	Similar to the format shown Figure 7.

Function	Communication during generation.
Query	PILOT queries Publisher for other data. The queries are in the form of Publisher?action=runSearch&search=commonSpacesForABuildingType &buildingType=BaltimoreRowHouse
Case	**Xml response**
	<results search="commonWallAssemblies"> <characteristics> <characteristic type="ExteriorWall" name="baltRowExteriorWall"> <Width> <HasMaxFeet>24</HasMaxFeet> <HasAvgFeet>18</HasAvgFeet> <HasMinFeet>12</HasMinFeet> </Width> </characteristic> </characteristics> </results>

Function	PILOT posts the generated interior layouts back to Publisher
Query	Pulisher?action=generationThreadTerminationReport& generationId=123& terminationStatus=successWithLayouts PILOT informs Publisher that a particular generation thread terminates as well as its termination status, so that Publisher can initiate query for the generated results.
Case	**Xml response**
	No response really needed.
Query	PILOT?action=nextLayoutResultRequest&generationId=123 Publisher queries for the next generated interior layout. (This procedure follows the enumeration model.)
Cases	**Xml response**
There is a next layout	<response status="success"> <msg>…</msg> <layout found="T" id="3"> <!--xml layout here--> </layout> </response>
There is no next layout.	<response status="success"> <msg>…</msg> <layout found="F" id="-1"> <!--empty--> </layout> </response>
Error	<response status="fail">

A generation cycle starts with a generation request from Publisher. PILOT dispatches a separate thread for each generation request so that multiple generation requests can be handled. Once a thread is dispatched, PILOT will send back status information immediately, as it may take the generation thread a while to complete the generation. Each thread is capable of conducting the standard query-and-response communications with Publisher individually until it terminates. There are three ways by which each thread possibly terminates: i) no errors with layouts generated, ii) no errors but no layout generated, and iii) errors occurred during execution. To handle possible error situations, responses are distinguished as success or failure: if success, a *found* tag is used to distinguish layouts generated or not; if fail, an *msg* tag contains the error message. Once the Publisher receives the successful termination status, it can start to retrieve the generation results by querying. It is possible that multiple layout results can be generated. Therefore, the procedure of layout result query follows an enu-

meration model; the publisher will keep query until there is no more layouts to send back. Table 1 gives a summary of the XML protocol adopted.

Generation and visualization

Generation process consists of two main steps, first being the decomposition of input footprint into a minimum set of rectangular blocks, and the second being assigning and organizing rooms. In the case of the Baltimore rowhouse, the initial layout estimation happens to be the same as the first few steps of the shape rules. This initial layout estimation can be used as the starting point for the further shape rule application without tree pruning. Shape grammar for Baltimore rowhouse consists of fifty two rules applied sequentially, where every rule is required or optional. The reason for sequential process is that rules are performed in eight phases, and the set of applied and optional rules determines the design outcome. The phases include block generation, space generation, stair generation, fireplace generation, space modification, front exterior feature generation, middle and back exterior feature generation, and interior feature generation.

The initial layout estimation is converted into graph-like data structures for further refinement by shape rules. The main manipulations are to refine the layouts from initial estimation, such as adding staircases, fireplaces, and interior doors. This requires basic functions, such as finding a shared wall of two rooms. Shape rules are further applied to add more details, such as interior doors, staircases, and openings. Figure 6 shows a screenshot of the PILOT system. There are three windows: the left is the generation window consisting of a tree of shape rule application (left panel) and a display of the layout (right panel), the middle window depicts the layout truth, and the right window shows the feature inputs.

The final step of the process is visualization of the generated layout as a three-dimensional model, which is achieved by parametric modeling. generated layout is captured in XML, as seen in Figure 8, and sent to Thetus Publisher for storing.

The modeling system uses an XML file as input to create a three-dimensional model of the building using features and parameters. Dimensions and variables are linked to geometry in such a way that when the parameter values change, the geometry updates accordingly. Based on the inter-element relationships stored in the lightweight model, the visualization modeling system determines how elements need to be created or updated. Upon creating a visual representation shown in Figure 9, the user is able to examine each part of a 3D model and rendering. The application is mainly

for conceptual design, but intended to be developed further so that it may become scalable to continue in detailed design.

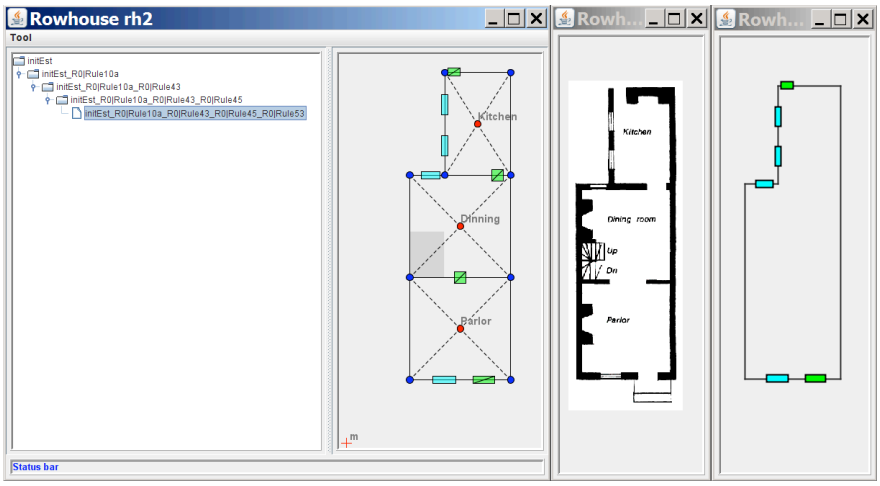

Fig. 6. Screenshot of the PILOT system
Left: generation window. Middle: layout truth. Right: Feature inputs.

Figure 7 shows results for generation of particular buildings and the original floor plan.

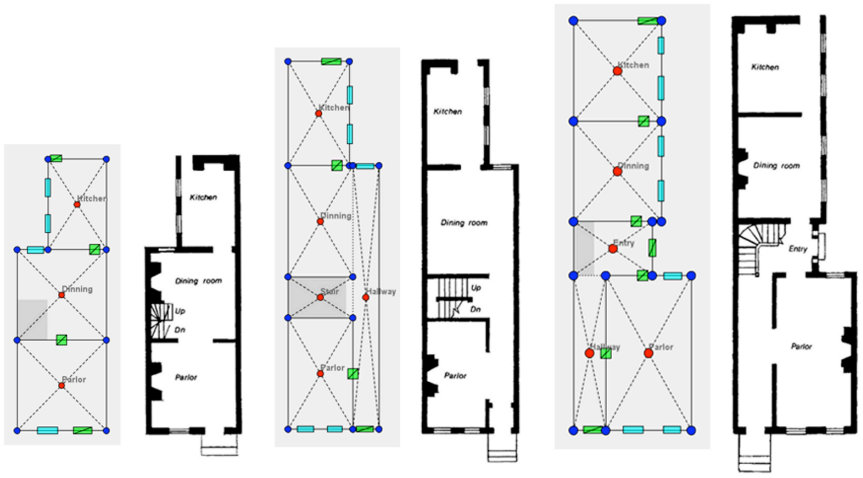

Fig. 7. Results of layout generation

```
<?xml version="1.0" encoding="UTF-8" ?>
- <building>
- <feature id="0" type="footprint" subtype="computation">
- <geometry>
- <polyline>
<point x="1.0250" y="338.0659" z="0.0000" />
<point x="1.0250" y="465.1029" z="0.0000" />
<point x="137.6360" y="465.1029" z="0.0000" />
<point x="137.6360" y="365.4787" z="0.0000" />
<point x="211.8517" y="365.4787" z="0.0000" />
<point x="211.8517" y="486.2189" z="0.0000" />
<point x="286.0674" y="486.2189" z="0.0000" />
<point x="286.0674" y="422.8709" z="0.0000" />
<point x="360.9599" y="422.8709" z="0.0000" />
<point x="360.9599" y="-6.0000" z="0.0000" />
<point x="-6.0000" y="-6.0000" z="0.0000" />
<point x="-6.0000" y="338.0659" z="0.0000" />
</polyline>
</geometry>
</feature>
- <feature id="1" type="footprint" subtype="display">
- <geometry>
- <polyline>
<point x="1.0250" y="338.0659" z="0.0000" />
<point x="1.0250" y="465.1029" z="0.0000" />
<point x="137.6360" y="465.1029" z="0.0000" />
<point x="137.6360" y="365.4787" z="0.0000" />
<point x="211.8517" y="365.4787" z="0.0000" />
<point x="211.8517" y="486.2189" z="0.0000" />
<point x="286.0674" y="486.2189" z="0.0000" />
<point x="286.0674" y="422.8709" z="0.0000" />
<point x="360.9599" y="422.8709" z="0.0000" />
<point x="360.9599" y="-6.0000" z="0.0000" />
<point x="-6.0000" y="-6.0000" z="0.0000" />
<point x="-6.0000" y="338.0659" z="0.0000" />
</polyline>
</geometry>
</feature>
```

Fig. 8. Example of generated layout in XML

The prototype application in Figure 9 also shows a way to modify features and parameters of each building element through window frames with parametric values. Thus, the parameters of each building element such as heights, widths, and lengths of building elements can be selected to modify and deliver a more customized representation of 3D building model. Volume of each space can be measured inside the parametric model as shown at the left corner of the figure.

Conclusion

This paper discusses interaction between knowledge-based and generative systems, outlining the complementary nature and the method for communication. Shape grammars contain rules for transforming geometrical entities, while knowledge-based model contains information about particular building types and context. Through exchange of information and query,

contextual aspects are explored and design knowledge is utilized to create building layouts. Baltimore rowhouse was presented as a particular case study expressing the process, knowledge acquisition, processing, characterization and visualization.

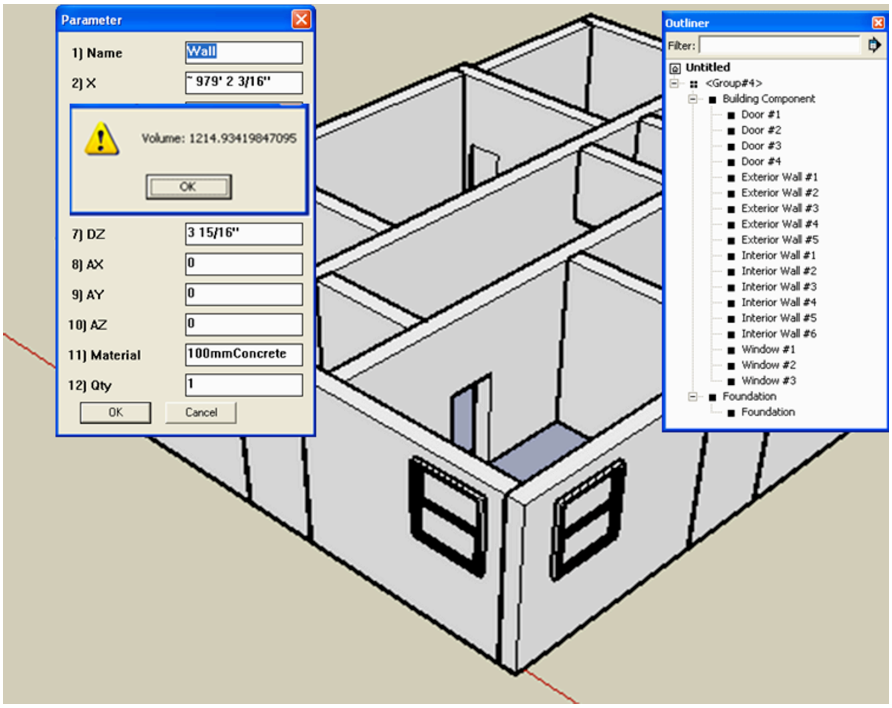

Fig. 9. 3-D Parametric Model

The discussed methodology for interaction between knowledge-based and generative systems is effective, and produces the desired results. The primary advantage of this method is that interaction is achieved by specifying the building type, so that the overall knowledge is modularized accordingly. The presented case study is a relatively simple building type, whose characteristics can be expressed relative to the major element, or firewall. The knowledge base contains information that is sufficient for shape grammar to perform selection of rules and generation of interiors. However, regional and cultural differences have not been tested yet, such as for Philadelphia or English rowhouses. Moreover, complex building types, such as multi-story mixed-use buildings or industrial facilities, are more difficult to characterize and require substantial descriptions. This process has not been performed for such intricate building types.

Currently, communication between the two systems is achieved by querying predefined searches. Constructing impromptu queries from the shape grammar side is not performed. Further research is needed to investigate methodology for dynamically accessing knowledge-base. Future plans include extending shape grammar rules to more building types, as well as knowledge-base model, and testing against regional and cultural differences.

References

1. Gero JS, Maher ML (1993) Modeling creativity and knowledge-based creative design. Lawrence Erlbaum Associates, Hillsdale
2. Kalay Y (2004) Architecture's new media: principles, theories, and methods of computer-aided design. MIT Press, Cambridge
3. McCullough M, Mitchell WJ, Purcell P (1990) The electronic design studio: architectural knowledge and media in the computer era. MIT Press, Cambridge
4. Mitchell WJ (1990) The logic of architecture: design, computation, and cognition. MIT Press, Cambridge
5. Stiny G, Mitchell WJ (1978) The Palladian grammar. Environment and Planning B: Planning and Design 5: 5-18
6. Chiou SC, Krishnamurti R (1995) The fortunate dimensions of Taiwanese traditional architecture. Environment and Planning B: Planning and Design 22: 547-562
7. Cagdas G (1996) A shape grammar: the language of traditional Turkish houses. Environment and Planning B: Planning and Design 23(4): 443-464
8. Duarte JP (2005) Towards the mass customization of housing: the grammar of Siza's houses at Malagueira. Environment and Planning B: Planning and Design 32: 347-380
9. Knight TW (1992) Designing with grammars. in Hatch (ed), Computer-Aided Architectural Design. Van Nostrand-Reinhold, New York
10. Nelson CB, Fenves SJ (1990) Manipulating shape and its function. Journal of Computing in Civil Engineering 4(3): 221-238
11. Grobler F, Kim H (2007) Ontology of a building to support reasoning in design process. Proceedings of International Workshop on Computing in Civil Engineering, ASCE, Pittsburgh
12. Grobler F, Aksamija A (2007) Architectural ontology: development of machine-readable representations for building design drivers. Proceedings of International Workshop on Computing in Civil Engineering, ASCE, Pittsburgh
13. Hayward ME, Belfoure C (2005) The Baltimore rowhouse. Princeton Architectural Press, New York