Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2010), *24*, 3–16.
Copyright © Cambridge University Press, 2010. 0890-0604/10 $25.00
doi:10.1017/S0890060409990138

Integration of knowledge-based and generative systems for building characterization and prediction

AJLA AKSAMIJA,[1] KUI YUE,[2] HYUNJOO KIM,[3] FRANCOIS GROBLER,[4] AND RAMESH KRISHNAMURTI[5]

[1]Tech Lab, Perkins+Will, Chicago, Illinois, USA
[2]School of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[3]Department of Civil and Environmental Engineering, California State University, Fullerton, California, USA
[4]US Army Corps of Engineers Construction Engineering Research Laboratory, Champaign, Illinois, USA
[5]School of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

(Received October 16, 2008; Accepted July 3, 2009)

Abstract

This paper discusses the integration of knowledge bases and shape grammars for the generation of building models, covering interaction, system, and implementation. Knowledge-based and generative systems are combined to construct a method for characterizing existing buildings, in particular, their interior layouts based on exterior features and certain other parameters such as location and real dimensions. The knowledge-based model contains information about spatial use, organization, elements, and contextual information, with the shape grammar principally containing style rules. Buildings are analyzed and layouts are generated through communication and interaction between these two systems. The benefit of using an interactive system is that the complementary properties of the two schemes are employed to strengthen the overall process. Ontologies capture knowledge relating to architectural design principles, building anatomy, structure, and systems. Shape grammar rules embody change through geometric manipulation and transformation. Existing buildings are analyzed using this approach, and three-dimensional models are automatically generated. Two particular building types, the vernacular rowhouse and high-rise apartment building, both from Baltimore, Maryland, are presented to illustrate the process and for comparing the utilized methodologies.

Keywords: Building Information Modeling; Knowledge-Based Model; Ontology; Shape Grammar

1. INTRODUCTION

The nature of architectural design poses immense challenges for computing and information processing in automated or semiautomated systems. Architectural design knowledge, thinking, and process are crucial components in the overall course of creating buildings with their computational representation of central concern (McCullough et al., 1990; Mitchell, 1990; Gero & Maher, 1993; Kalay, 2004). In particular, the types of information that architects seek vary depending on the nature of the design problem; however, the method by which information is sought is often ambiguous. Computational analyses of sources of ideas and design require a thorough understanding and comprehension of intentions, as well as contextual aspects. In the work reported in this paper, knowledge-based and generative systems are combined to construct a method for the analysis of building types.

Reprint requests to: Francois Grobler, US Army Corps of Engineers Construction Engineering Research Laboratory, P.O. Box 9005, Champaign, IL 61826, USA. E-mail: Francois.Grobler@erdc.usace.army.mi

A knowledge-based model represents knowledge about a subject, describing individuals as basic objects, classes as collections or types of objects, properties and characteristics, and relations between objects. In this work, ontology is used as the knowledge-based model to capture architectural design principles. It contains information about building designs, location, use, orientation, and size, but does not give form to buildings with geometric meaning. A shape grammar is a rule-based formalism for producing designs or the generation of geometric shapes (Stiny, 1980, 2006). There are certain similarities between a shape grammar and a knowledge-based model, mainly in that both contain design rules, although the nature of the rule varies. For ontology, rules represent standard logical mechanisms for extracting new knowledge from asserted knowledge. For shape grammars, rules represent compositional entities, embodying change through geometric manipulation and transformation.

Two kinds of shape grammars have been identified: analytical and original (Knight, 1991). Analytical shape grammars were developed to analyze and describe historical styles, or designs by specific architects (Stiny & Mitchell, 1978; Chiou & Krishnamurti, 1995; Cagdas, 1996; Duarte, 2005a). They use a set of existing designs, as a corpus, to develop the design language and infer shape rules. Grammars are tested by using the rules to generate designs both in the corpus and new. In contrast, original shape grammars are based on rules intended to create instances of new (or original) designs. These types of grammars are widely considered as *not* analytical, owing to the difficulty of "translation of abstract, experimental form into architectural designs that fit particular design contexts" (Knight, 1991). The shape grammars considered in this paper are analytical.

The combinatory nature of design rules encapsulated by a shape grammar and ontology offers the possibility that design knowledge can be explicitly represented, maintained, and processed. Architectural design knowledge is captured in the ontology and processed by a shape grammar, thus allowing for generation and analysis. In this sense, rules can be customized according to context, building size, style, or function.

In many analytical shape grammars, found in the literature, rule descriptions of the form "if the back or sides are wide enough, rule 2 can be used . . ." are commonplace, but such rules are equally inherently countercomputable. Exceptions to this, in a limited way, can be found in certain mechanical engineering shape grammars (McCormack & Cagan, 2002; Pugliese & Cagan, 2002; McCormack et al., 2004). 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 could benefit from the assistance of the ontology. In our attempts to quantify shape rules (Yue & Krishnamurti, 2008), originally specified in the traditional way, we frequently found that the only way to distinguish certain rules is to employ threshold values statistically derived from a building sample, for example, the area of a space for a particular use. The ontology can provide such values dynamically as new building samples are added.

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

2. METHODOLOGY: PROCESS AND COMPONENTS

The starting point of the interaction focuses on requirements, including building location, dimensions, and functional type. Figure 1 presents the overall process and interaction. A "light-weight" parametric building information model (BIM) is created based on the requirements implied by what is known about the building; this model contains information about the building shell, general building anatomy, and components. This data is also the initial input for the shape grammar rules, although additional information about the building, such as typical spaces and dimensions, are also needed for the process. The means of providing such information for use by the shape grammar rules is through communication with several ontologies, which include specific information according to building type, location, environment, structural system, and context. Once the information is received, the shape grammar system selects rules and configures a spatial

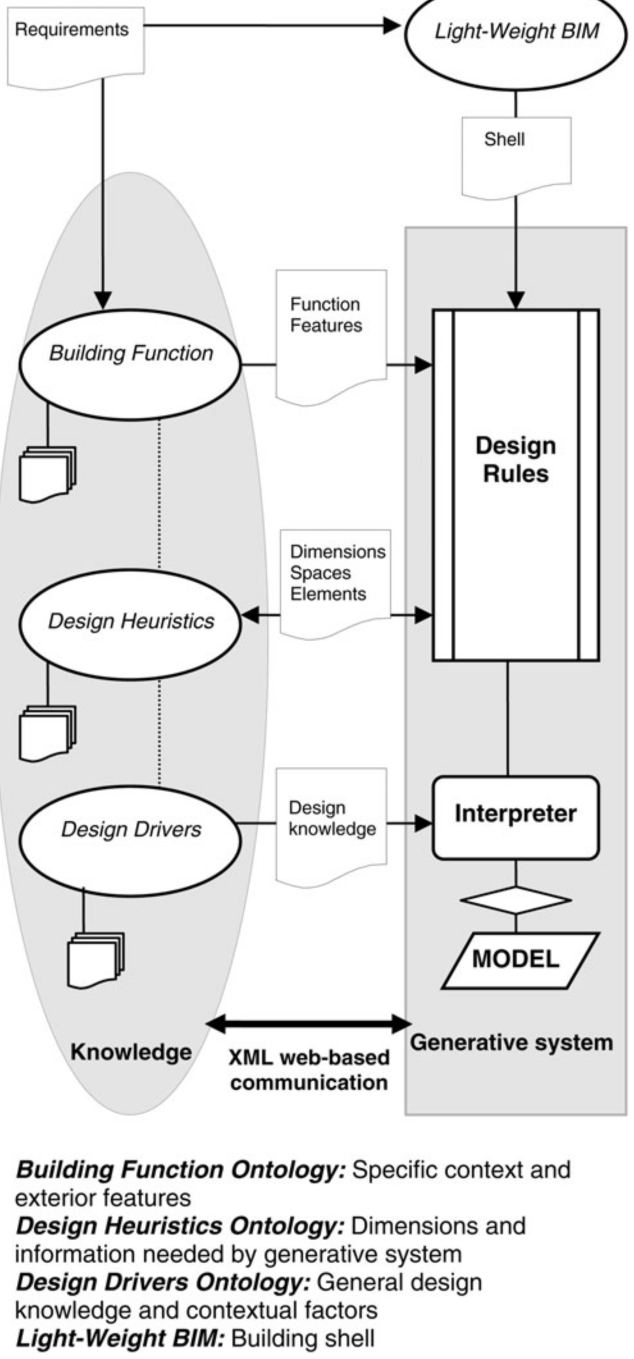

Requirements
Light-Weight BIM
Shell
Function Features
Building Function
Design Rules
Dimensions Spaces Elements
Design Heuristics
Design knowledge
Design Drivers
Interpreter
MODEL
Knowledge
XML web-based communication
Generative system
Building Function Ontology: Specific context and exterior features
Design Heuristics Ontology: Dimensions and information needed by generative system
Design Drivers Ontology: General design knowledge and contextual factors
Light-Weight BIM: Building shell

Fig. 1. The interaction and communication process between ontologies and shape grammar.

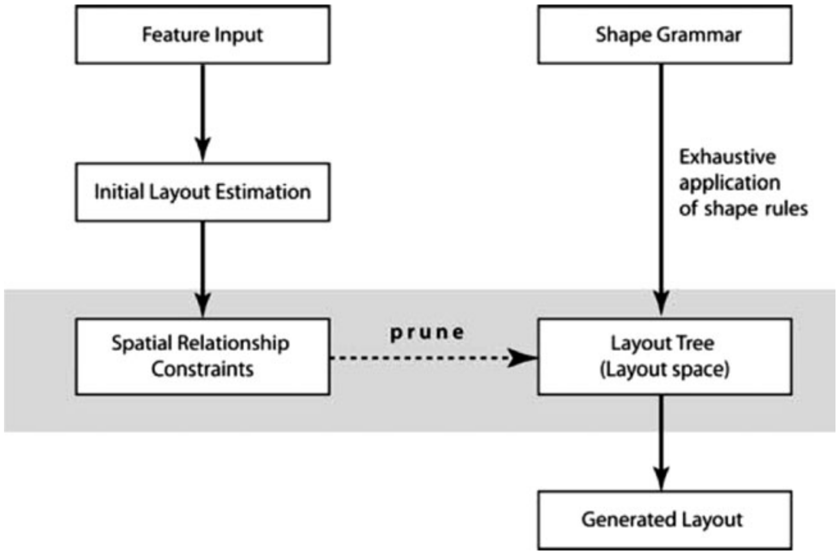

Feature Input
Initial Layout Estimation
Spatial Relationship Constraints
prune
Shape Grammar
Exhaustive application of shape rules
Layout Tree (Layout space)
Generated Layout

Fig. 2. Approach for layout generation.

organization. During the application of shape rules, the shape grammar system may query the ontology system for certain specific information, in particular, data and facts of a statistical nature. 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. The end result is a generated layout, outlining spatial organization. The information populates the parametric BIM and can be visualized in three dimensions.

3. ONTOLOGY STRUCTURE AND CONTENTS

The process for the analysis of building types and generation of building models relies on several ontologies that capture specialized knowledge. In this respect, four ontologies are used in the overall process: building function, building anatomy, design heuristics, and design drivers. Of these, building function, design heuristics, and design drivers ontologies are structured and contained within the same environment, whereas building anatomy is used to construct a lightweight representation of building features and to display/hide/erase the element(s) of a model and have an instant view of each element. The main objective of the building anatomy is to provide an understanding of the essential concepts of a building, offering a unique visual approach to the user.

The building function ontology consists of predefined searches that characterize use, depending on site context and exterior features. Building shell is displayed through the building anatomy ontology or lightweight BIM. It is used to visualize general building structure and elements without specific information about its interior spatial organization. The purpose of developing a lightweight BIM is to build a neutral, semantic, slimmed-down representation of a building, the objects in that building, and the relationships among them, thus capturing common building elements and anatomy (Kim & Grobler, 2007). 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.

Design heuristics and design drivers ontologies capture architectural knowledge and are used in conjunction with the shape grammar to generate building models (Aksamija & Grobler, 2007). It is a method for providing constraints

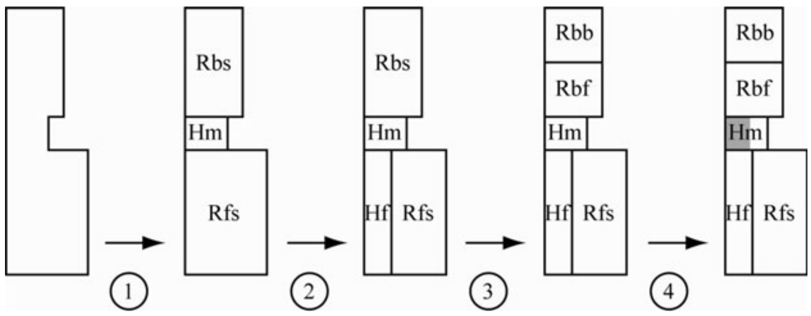
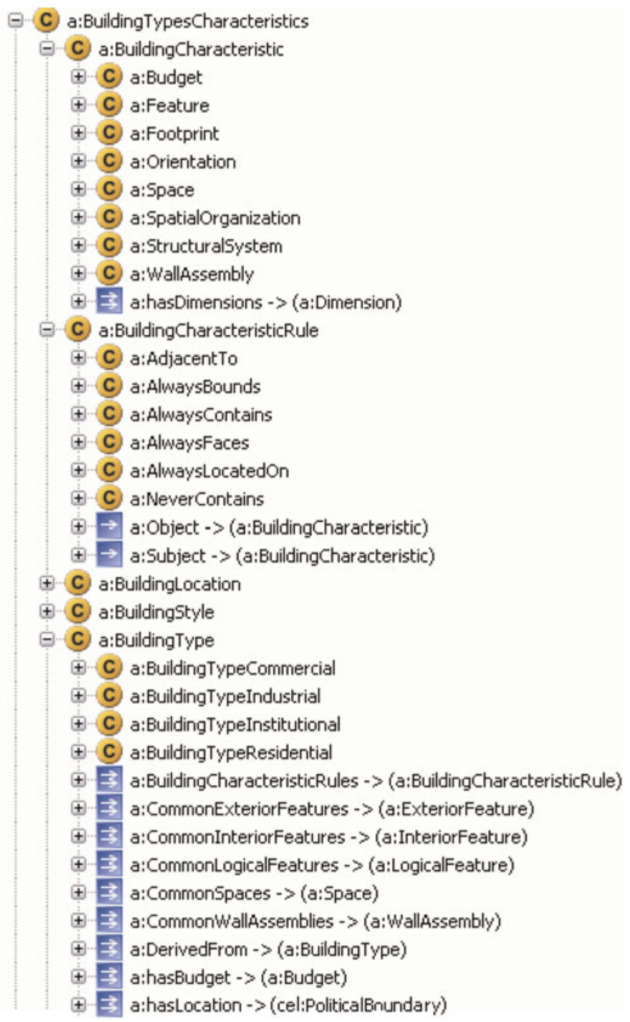


Fig. 3. A sample derivation.



```
a:BuildingTypesCharacteristics
    a:BuildingCharacteristic
        a:Budget
        a:Feature
        a:Footprint
        a:Orientation
        a:Space
        a:SpatialOrganization
        a:StructuralSystem
        a:WallAssembly
        a:hasDimensions -> (a:Dimension)
    a:BuildingCharacteristicRule
        a:AdjacentTo
        a:AlwaysBounds
        a:AlwaysContains
        a:AlwaysFaces
        a:AlwaysLocatedOn
        a:NeverContains
        a:Object -> (a:BuildingCharacteristic)
        a:Subject -> (a:BuildingCharacteristic)
    a:BuildingLocation
    a:BuildingStyle
    a:BuildingType
        a:BuildingTypeCommercial
        a:BuildingTypeIndustrial
        a:BuildingTypeInstitutional
        a:BuildingTypeResidential
        a:BuildingCharacteristicRules -> (a:BuildingCharacteristicRule)
        a:CommonExteriorFeatures -> (a:ExteriorFeature)
        a:CommonInteriorFeatures -> (a:InteriorFeature)
        a:CommonLogicalFeatures -> (a:LogicalFeature)
        a:CommonSpaces -> (a:Space)
        a:CommonWallAssemblies -> (a:WallAssembly)
        a:DerivedFrom -> (a:BuildingType)
        a:hasBudget -> (a:Budget)
        a:hasLocation -> (cel:PoliticalBoundary)
```

Fig. 4. Contents of design heuristics ontology. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

to the shape grammar rules, which are needed to specialize solutions. Shape grammars perform transformations of geometrical objects based on information provided by ontologies. The design heuristics ontology was specifically designed to respond to shape grammar questions and structured to contain information for specific building types. Communication is performed through XML Web services and queries. The building function and design heuristics ontologies are integrated through specific examples of buildings. Generalized information about a building type is contained in design heuristics, whereas particular building instances are stored in the function ontology. Moreover, building functions, such as residential, commercial, industrial, and educational, are incorporated into both ontologies. The design heuristics ontology contains information about almost 300 different building types and characterizes the typology of these types by describing features, properties, design rules, common spaces, and spatial organization. The purpose is to interact with the shape grammar interpreter in two ways: first, by using building type to select the appropriate shape grammar, and second, by providing appropriate values and properties for use by the shape rules as and when needed.

4. SHAPE GRAMMAR AND LAYOUT GENERATION

The ability to "predict" the interior layout of a building from its exterior and surrounding features has a number of practical applications. For instance, accessing the environmental impact of demolition and salvage of building stock requires one to estimate the amount of renewable materials (Lund & Yost, 1997). Automating the process of interior layout prediction would greatly assist in this process.

Although it is not especially hard for a human to roughly estimate building layouts from familiar features, programming a machine to do so is much more difficult. However, by using knowledge of building styles this task or a subset of tasks can 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 (Stiny, 1980) offer the facility of capturing the spatial and topological aspects of building styles. 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, we seek an algorithm to determine the interior layout of a building given an input of building features visible exteriorly and a shape grammar that describes the building style. The building feature input includes the footprint of each story, as well as reasonably complete exterior features, for example, windows, chimneys, and surrounding buildings.

In principle, when applied exhaustively, shape grammars generate, as a tree, the entire layout space of a style. The desired layouts are those satisfying the constraints posed by the feature input. However, such constraints are typically specified by the feature input implicitly, which is difficult to apply; it is necessary to process the feature input so that the directly applicable constraints can be extracted. Figure 2 demonstrates the approach. As shown in the figure, we employ a step, named initial layout estimation, that derives a preliminary incomplete layout from the feature input. From this estimate, 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.

5. COMMUNICATION BETWEEN ONTOLOGY AND SHAPE GRAMMAR

To demonstrate the process and communication between the knowledge bases and generative systems, two distinct building types are used as exemplars: the vernacular rowhouse and high-rise apartment building, both from the city of Baltimore, Maryland. Although their shape grammars and types of information that these utilize differ for the two exemplars, the

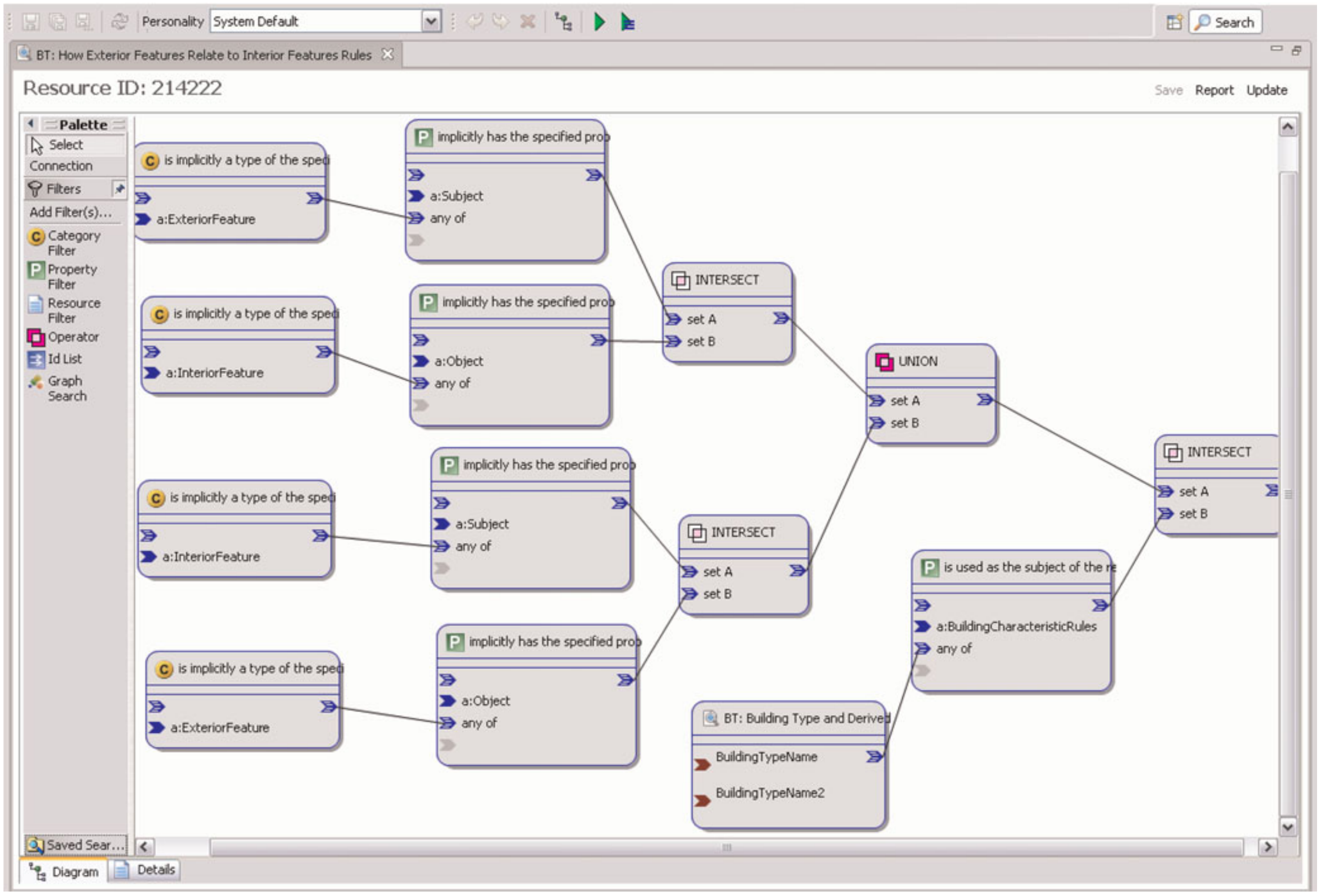


Fig. 5. Queries constructed based on shape grammar questions. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

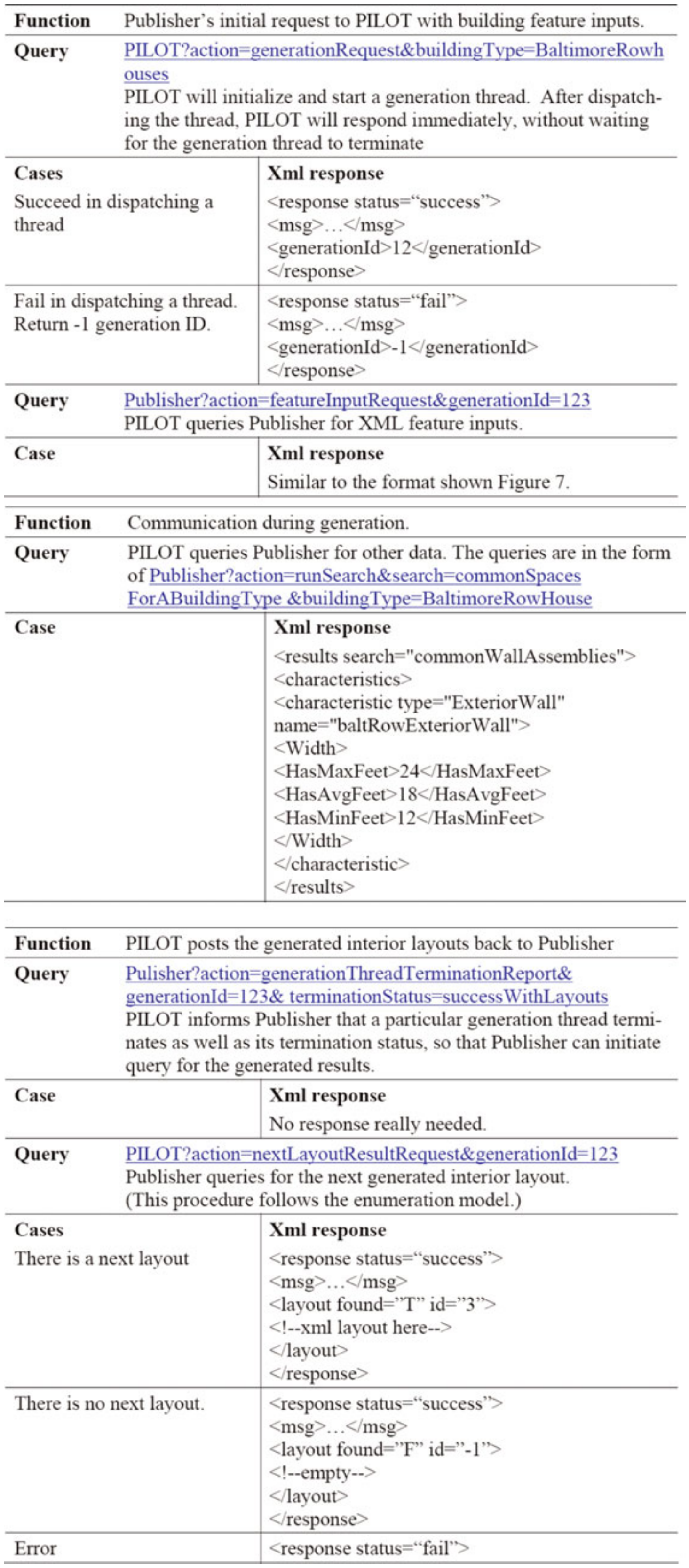
process is similar. Ontologies contain descriptions of general characteristics, organized by specific building types. For ease of implementation, the types of queries that shape grammar algorithms pose are predefined, as will be presented in the subsequent examples; pending further research, dynamically generated queries could also be handled in the same manner. The responses from ontologies are dynamic and change according to the specified building type. Communication is achieved through Web-based XML interaction, where semantic knowledge can be utilized for querying specific questions or design rules associated with certain building types.

5.1. The Baltimore rowhouse

The Baltimore rowhouse is typically quite narrow, two stories high, and oriented along a north–south or east–west axis. Living rooms typically face the front and are directly accessible from the street or narrow hallway when there are two bays, and the kitchen is located in the back (Hayward & Belfoure, 2005). Wood stairs are often a single run and are oriented along one firewall. The firewalls are primarily constructed of brick, with a wood framed structure for interior partitions, floors, and roofs. Roofs typically have nominal slope.

The information presented above is captured in the design heuristics and design drivers ontologies through different methods. For example, the general building class contains elements in which case an instantiation of a Baltimore rowhouse presents actual elements of this particular building type. For instance, the firewall is a key element of the rowhouse; its spatial organization is always linear and dependent on this key element. Similarly, building spaces belonging to a 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.

The Baltimore rowhouse shape grammar consists of 52 shape rules applied in sequence, where each rule is either required or optional. The reason for the sequential process is that rules are grouped into eight phases, and the set of applied and optional rules determines the design outcome. The phases are block generation, space generation, stair generation, fire-



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=commonSpaces ForABuildingType &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> </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">

Fig. 6. XML communication protocol. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

```
Input
footprint:
    the building footprint
pipes:
    a list of ventilation pipes
candidateUnits:
    a list of parameterized candidate layout units
constraints:
    constraints a candidate layout must satisfy(, currently including no
    overlapping constraint,  inside footprint constraint, as well as window-
    sides on footprint constraint)

Output
layoutSolutions:
    layout solutions found


Algorithm
find all unit layouts, unitLayouts, which matches a pipe in pipes under
constraints
use unitLayouts to initiate a list of candidate layout solutions,
candidateSolutions
push candidateSolutions into a stack, candidateSolutionStack

while candidateSolutionStack is not empty
    pop a candidate solution
    if all pipes of the candidate solution have been covered by apartment units
       add it to layoutSolutions
    else
       find all possible layout units matches a pipe in the uncovered pipes
       if none found
           continue
       else
           create new candidate layout solutions and push into stack

return layoutSolutions
```

Fig. 7. High-rise prediction algorithm.

place generation, space modification, front exterior feature generation, middle and back exterior feature generation, and interior feature generation. See Yue and Krishnamurti (2008) for more details and visual examples of the Baltimore rowhouse shape grammar and its rules, their parameters, their coding, and data structures.

The starting point for the interaction between the shape grammar and the ontology is a list of questions that a shape grammar poses to the knowledge-based system. For example, questions on a Baltimore rowhouse relate to building orientation, surrounding context, spaces, dimensions, and construction method. Examples of such queries are the following:

- 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?
    — The living room 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, although 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?

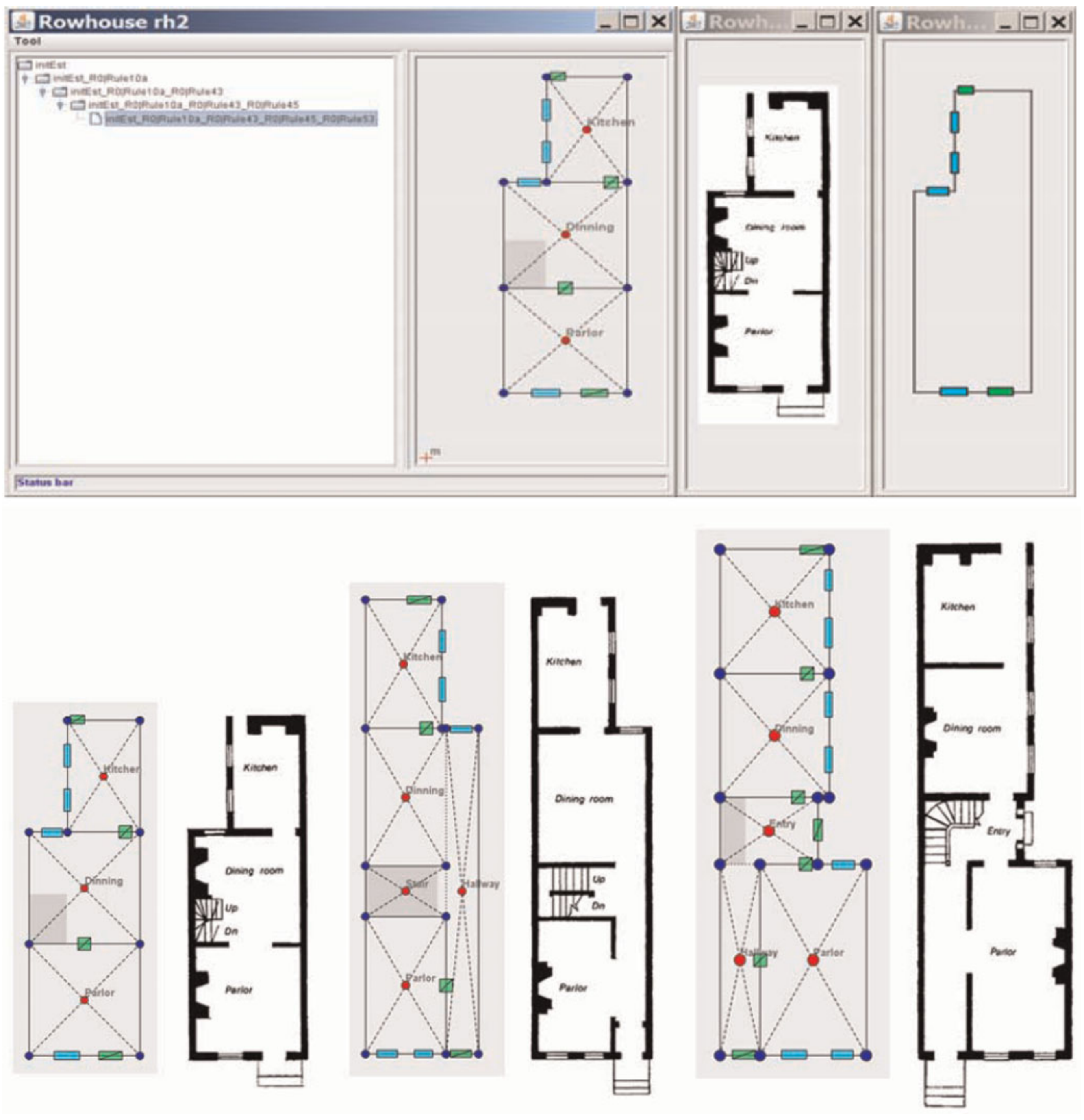

Rowhouse rh2
Tool
initEst
initEst_R0|Rule10a
initEst_R0|Rule10a_R0|Rule43
initEst_R0|Rule10a_R0|Rule43_R0|Rule45
initEst_R0|Rule10a_R0|Rule43_R0|Rule45_R0|Rule53
Status bar
Rowh...
Kitchen
Dinning
Parlor
Dining room
Up
Dn
Stair
Hallway
Entry

Fig. 8. PILOT system for Baltimore rowhouses and tested layouts. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

— For fireplace (depth), staircase (slope), staircase (run), parlor (width), parlor (depth), hallway (width), hallway (depth), interior wall (thickness), and so forth

- Do features align with stories?
- Is a group of features symmetric? If so, what is the symmetric axis?

As the shape grammar rules start to execute, the system poses queries to the ontology, such as the required or optional features, required spaces, dimensions, and orientation. This information is received, processed, and then, according to the responses, subsequent rules are implemented.

The need for querying the ontology is decided by the particular nature and/or design of a shape rule. In some cases, the application of some shape rules relies upon the information queried from the ontology. In some others, it does not. For instance, the sequence of shape rule application in Figure 3 starts from a footprint, which comes from the feature input. Step 1

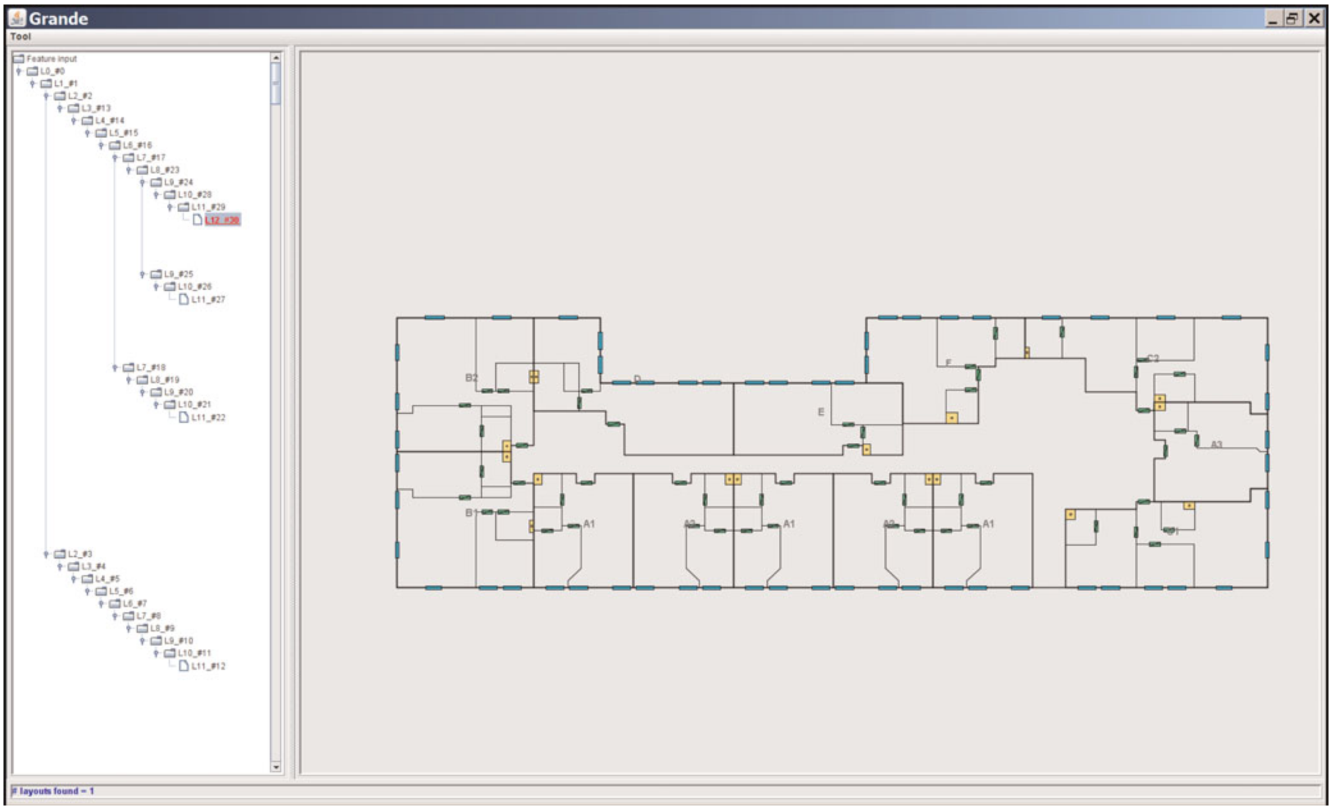


Fig. 9. PILOT system for high-rise apartment building. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

decomposes the footprint into front, middle, and back blocks based on the shape of the footprint boundary. As a result, this shape rule does not need to query the ontology. Step 2 needs to cut off a hallway from the front block (*Rfs*). Whether this particular shape rule is applicable depends on several factors, among them, the width and area of the front block. Such threshold values are better queried from the ontology, rather than hardcoded in the shape rules. This is particularly necessary when new building samples are added into the ontology in a progressive manner. Thus, a query is devised here. It is instructive to note that the mechanism described here differs from Duarte (2005b), where the knowledge base is used to infer the description of the interior layout of a building; the shape grammar is then executed, based on the description obtained. Here, the communication with the knowledge base is based on the needs of each shape rule. Moreover, the shape rules are so designed to be extensible to cater for situations when new building samples have been added.

Interaction between shape grammar and ontology is accomplished through XML communication by Web services. Thetus Publisher[1] contains the ontologies discussed in this paper. 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 knowledge is discovered. In the layout generation process, the queries are performed by the shape grammar system, where specific questions are asked from the ontology. The questions are directed to Thetus, which gathers the necessary information and sends replies to the shape grammar rules. Figure 4 presents the contents of the design heuristics ontology for a Baltimore rowhouse. A constructed search for the relationship between exterior and interior features is shown in Figure 5. Once the information is received, the generation process initializes. The generated layouts are sent back to Thetus Publisher as XMLBIMs, where they are stored.

There are technical issues associated with implementing communication through Web services. The generative system, named PILOT (Proposing Interior Layout Over building Types), poses queries to the ontology system. The basic model of Web services is query and response; that is, one side starts an 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. Because of the limitations of the query and response model, certain communications have to be realized by multiple query and responses. 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 awhile to

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

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 can possibly terminate: with no errors and layouts generated, with no errors but no layout generated, and with errors occurring during execution.

To handle possible error situations, responses are distinguished as "success" or "fail" through the use of tags: if success, a *found* tag is used to distinguish layouts generated or not; if fail, the *msg* tag contains an error message. Once Publisher receives a 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 results query follows an enumeration model; Publisher will keep a query until there are no more layouts to send back. Figure 6 gives a summary of the XML protocol adopted.

5.2. High-rise apartment building

High-rise apartment buildings are more complex examples than the Baltimore rowhouse. The apartment building poses a challenge, namely, it is not feasible or appropriate to capture the entire layout of a floor by using a single shape grammar. In contrast, it is not difficult to develop a shape grammar for apartment units, in a manner similar to that for the Baltimore rowhouse, so that all possible layouts of apartment units can be generated. Even assuming that possible apartment units are available, a different set of features has to be utilized so that the entire layout can be "assembled" from possible apartment units. High-rise apartments usually have a uniform facade, which posits very weak constraints on the interior layouts. Therefore, it is hard to develop possible interior layouts based directly on facade features. However, equipment pipes over the roof can be utilized, as these indirectly reflect the interior arrangement. Bathroom ventilation pipes can be observed on the roof, together with other HVAC components. Following this analysis, using knowledge of different possible unit layouts, the prediction algorithm (Fig. 5) is devised as a search for reasonable arrangements of different units by aligning them with pipes with various possibilities and eliminating any unreasonable solution using the constraints.

To generate possible apartment units from a shape grammar, communication similar to that for the Baltimore rowhouse grammar is required. In addition, possible positions for the pipes in a unit are also queried from the ontology; a shape grammar typically does not record such information. To implement the high-rise prediction algorithm, as shown in Figure 7, it is important to resolve whether an apartment unit matches a pipe. Other factors also need to be considered, such as locating the sides for the placement of windows and fitting the building footprint. Moreover, the determinant factors may vary from one high-rise building to another.

There are three type constraints used in the implementation: window-side constraints, no apartment unit overlap constraints, and inside the building footprint constraints. All three

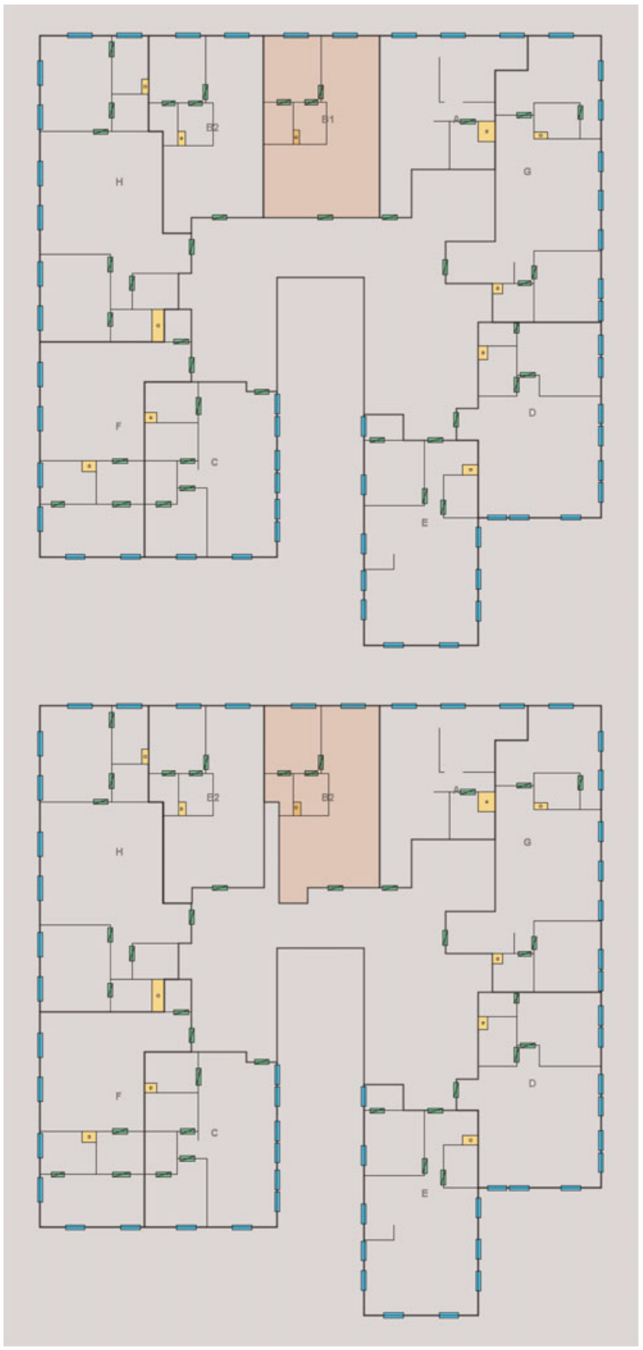
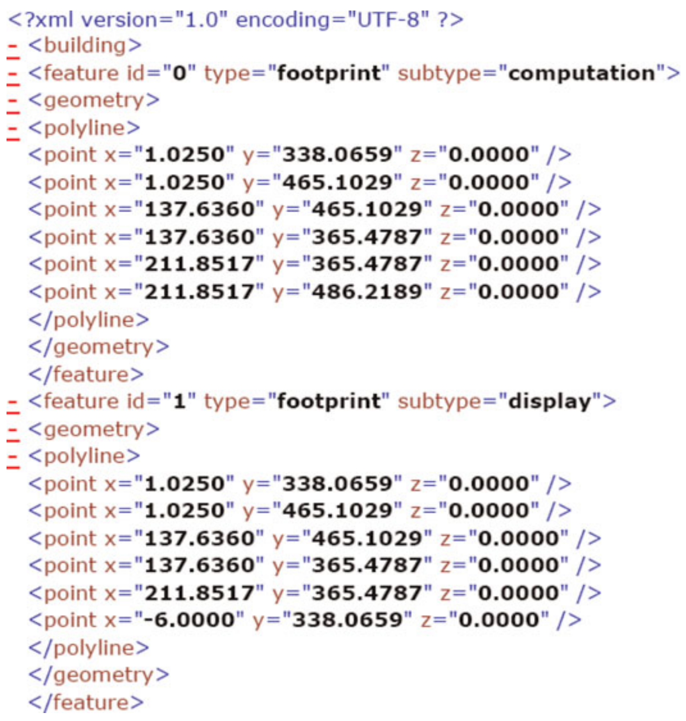


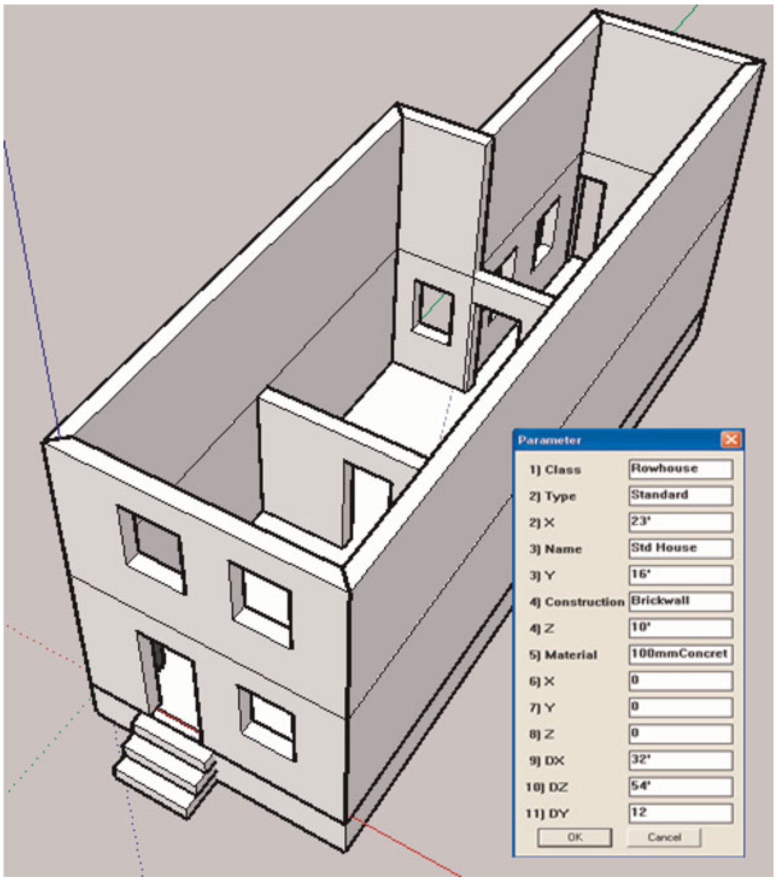
Fig. 10. Results of interior generation for high-rise apartment building. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

constraints can be implemented in terms of Boolean operations. Window-side constraints ensure that apartment units have enough walls facing the building exterior so that natural lighting can be provided. This constraint is implemented by testing whether the bounding boxes for the windows intersect the polygon representing the building footprint. As the name implies, the no apartment unit overlap constraints ensure that two apartment units do not overlap. This is implemented by testing whether the boundaries of two units intersect. Inside



```
<?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" />
  </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="-6.0000" y="338.0659" z="0.0000" />
  </polyline>
  </geometry>
  </feature>
```

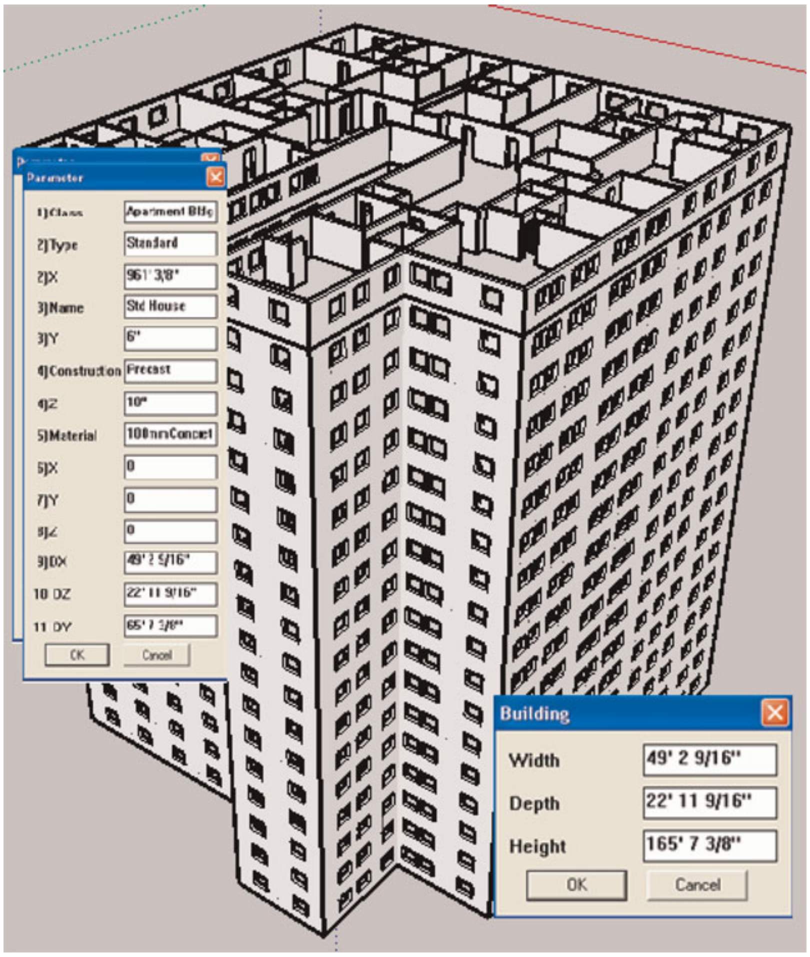
Fig. 11. XML capturing building layout. [A color version of this figure can be viewed online at journals.cambridge.org/aie]



Parameter

1] Class	Rowhouse
2] Type	Standard
2] X	23'
3] Name	Std House
3] Y	16'
4] Construction	Brickwall
4] Z	10'
5] Material	100mmConcret
6] X	0
7] Y	0
8] Z	0
9] DX	32'
10] DZ	54'
11] DY	12

Fig. 12. Baltimore rowhouse model. [A color version of this figure can be viewed online at journals.cambridge.org/aie]



Parameter

1)Class	Apartment Bldg
2)Type	Standard
2)X	961' 3/8"
3)Name	Std House
3)Y	6"
4)Construction	Precast
4)Z	10"
5)Material	100mmConcret
6)X	0
7)Y	0
8)Z	0
9)DX	49' 2 5/16"
10 DZ	22' 11 9/16"
11 DY	65' 7 3/8"

Building

Width	49' 2 9/16"
Depth	22' 11 9/16"
Height	165' 7 3/8"

Fig. 13. High-rise apartment building model. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

the building footprint constraints ensure that no apartment units fall outside the building footprint. This can be tested for by counting how many of points on the boundary of an apartment unit fall outside the building footprint.

6. GENERATION AND VISUALIZATION

The generation process consists of two main steps: the first is the decomposition of the input footprint into a minimum set of rectangular blocks, and the second is assigning and organizing spaces. In the case of the Baltimore rowhouse, the initial layout estimation happens to be identical to the first few steps in applying the shape rules. This initial layout estimate can be used as the starting point for further shape rule application without requiring tree pruning.

The initial layout estimate is converted into graphlike data structures for further refinement by the shape rules. The main manipulations correspond to layout refinement from the initial estimate by incrementally adding internal features, for example, staircases, fireplaces, and interior doors. This requires certain basic functions such as finding the shared wall between two rooms. Shape rules are then applied to add more detail such as interior doors, staircases, and openings previously mentioned. Figure 8 shows a screenshot of the PILOT system, as well as several test results. There are three windows: the left-most is the generation window, which is split into two panels, consisting of a tree of shape rule application (in the left panel) and a display of the layout (in the right panel), the middle window depicts the layout truth, and the right-most window shows the feature inputs.

The shape grammar for high-rise apartment building types relies on apartment layouts and the position of mechanical systems, particularly ventilation pipes. Figure 9 shows the PILOT system for a high-rise apartment building. As this building type is much more complex, it is possible to have more than one layout for a certain building as shown in Figure 10.

The final step in the process is visualizing the generated layout as a three-dimensional model. A generated layout is captured in XML, as shown in Figure 11, 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 parameter values change, the geometry updates accordingly. Based on the interelement relationships stored in the lightweight model, the visualization modeling system determines how elements need to be created or updated.

Upon creating the visual representations, the user is able to examine each part of a three-dimensional model and rendering. The prototype application also shows a way of modifying features and parameters of each building element through window frames with parametric values as shown in Figures 12 and 13. Currently, a typical single floor layout of a building model is translated into a three-dimensional (3-D) model, but a further development is needed to build a 3-D visualization model of multiple floors in a building. 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 the 3-D building model. The volume of each space can be measured inside the parametric model as shown at the left corner of the figure.

7. CONCLUSION

This paper discusses the interaction between knowledge-based and generative systems, outlining their complementary nature and a method for communication. Shape grammars contain rules for transforming geometrical entities, whereas the 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 and high-rise apartment building types are presented as particular case studies expressing the process, knowledge acquisition, processing, characterization, and visualization.

The methodology discussed in this paper for the interaction between knowledge-based and generative systems is effective and produces desired results. The primary advantage of this method is that interaction is achieved by specifying a building type so that the overall knowledge can be modularized accordingly. The case studies presented reveal that knowledge is modularized according to building type, and this information is distinctly used to generate interior layout. 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 the methodology for dynamically accessing the knowledge base.

REFERENCES

Aksamija, A., & Grobler, F. (2007). Architectural ontology: development of machine-readable representations for building design drivers. *Proc. Int. Workshop on Computing in Civil Engineering*, pp. 168–175. Pittsburgh, PA: ASCE.

Cagdas, G. (1996). A shape grammar: the language of traditional Turkish houses. *Environment and Planning B: Planning and Design 23(4)*, 443–464.

Chiou, S.C., & Krishnamurti, R. (1995). The fortunate dimensions of Taiwanese traditional architecture. *Environment and Planning B: Planning and Design 22*, 547–562.

Duarte, J.P. (2005*a*). Towards the mass customization of housing: the grammar of Siza's houses at Malagueira. *Environment and Planning B: Planning and Design 32*, 347–380.

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

Gero, J.S., & Maher, M.L. (1993). *Modeling Creativity and Knowledge-Based Creative Design*. Hillsdale, NJ: Erlbaum.

Hayward, M.E., & Belfoure, C. (2005). *The Baltimore Rowhouse*. New York: Princeton Architectural Press.

Kalay, Y. (2004). *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*. Cambridge, MA: MIT Press.

Kim, H., & Grobler, F. (2007). Ontology of a building to support reasoning in design process. *Proc. Int. Workshop on Computing in Civil Engineering*, pp. 151–158. Pittsburgh, PA: ASCE.

Knight, T.W. (1991). Designing with grammars. In *Computer-Aided Architectural Design* (Schmitt, G.N., Ed.), pp. 33–48. Wiesbaden: Vieweg.

Lund, E., & Yost, P. (1997). Deconstruction—building disassembly and material salvage: the Riverdale case study. Upper Marlboro, MD: NAHB Research Center, Inc.

McCormack, J.P., & Cagan, J. (2002). Designing inner hood panels through a shape grammar based framework. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *16*, 273–290.

McCormack, J.P., Cagan, J., & Vogel, C.M. (2004). Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars. *Design Studies 25*, 1–29.

McCullough, M., Mitchell, W.J., & Purcell, P. (1990). *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*. Cambridge, MA: MIT Press.

Mitchell, W.J. (1990). *The Logic of Architecture: Design, Computation, and Cognition*. Cambridge, MA: MIT Press.

Pugliese, M.J., & Cagan, J. (2002). Capturing a rebel: modeling the Harley–Davidson brand through a motorcycle shape grammar. *Research in Engineering Design 13*, 139–156.

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

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

Stiny, G. (2006). *Shape: Talking about Seeing and Doing*. Cambridge, MA: MIT Press.

Yue, K., & Krishnamurti, R. (2008). A technique for implementing a computation-friendly shape grammar interpreter. In *Design Computing and Cognition '08* (Gero, J.S. & Goel, A.K., Eds.), pp. 61–80. New York: Springer Science + Business Media B.V.

Ajla Aksamija leads a Tech Lab at Perkins+Will as a Building Technology Researcher. She received a PhD in architecture from the University of Illinois at Urbana–Champaign, which focused on technology and the environment. Her professional experience includes the US Army Corps of Engineers ERDC Construction Engineering Research Laboratory, City of Champaign, National Institute for Urbanism in Bosnia-Herzegovina, and Doxat Architecture. Her research interests include computational design, emerging building technologies, and integrated design, as well as relationships between the environment and technology. She has received numerous awards, such as the Francis J. Plym Doctoral Fellowship in Architecture, the Edward L. Ryerson Traveling Fellowship, the Frank B. and Jennie B. Long Award, the White Prize in Architectural Practice, and first place in the design competition for the Champaign County Historical Museum Lot Project.

Kui Yue is a PhD candidate at Carnegie Mellon University. He received a BA in architecture from Tongji University and an MS in architecture from Mississippi State University. His research is in the field of shape grammars, in particular, their computational complexity, implementation, and applications to practice. His professional experience includes being a de-

signer at DDB International, Ltd., Shanghai, and an internship at SDET at Microsoft, Redmond, CA. He won the Young CAADRIAN award in 2007. He has worked on projects using laser scanning and embedded sensor technologies to identify defects on construction sites and using shape grammars to determine building interior layouts from exterior features.

Hyunjoo Kim is an Assistant Professor in the Department of Civil and Environmental Engineering at California State University. He previously worked at US Army CERL, focusing on BIM application in the area of CADD design collaboration and implementing on reasoning process. He received his PhD in 2002 from the University of Illinois at Urbana–Champaign with the dissertation "Knowledge Discovery and Machine Learning in Construction Project Databases." He was a Project Manager for CPM Construction, Inc., in 2002–2004 and a Senior Program Manager in the international construction project US Military Bases Relocation located in Korea in 2004–2006. His main research areas include the use of information technology for project management, BIM, artificial intelligence, and machine learning.

Francois Grobler is a Civil Engineer and Principal Investigator at the US Army Corps of Engineers ERDC Construction Engineering Research Laboratory. His professional career started as a structural designer for a large water utility company, followed by construction field work on a variety of construction projects, serving as the owner's representative. After completing graduate degrees he held teaching positions at the University of Illinois in the Civil Engineering Department and Penn State University in the Architectural Engineering Department. His research has focused on computer modeling of the built environment and decision support derived from such models. Grobler has been developing object-oriented data representations for construction related information since 1985, and in 1997 he joined the International Alliance for Interoperability (IAI) to broaden this effort. He has served as the Technical Coordinator for the IAI in North America since 1999.

Ramesh Krishnamurti is currently a Professor in the School of Architecture at Carnegie Mellon University where he directs the Graduate Program in Computational Design. He has degrees in electrical engineering, computer science, and systems design. He has previously taught and worked in Canada, the United Kingdom, and Taiwan. Dr. Krishnamurti is a shape grammarist; his main area of research focuses on the formal, semantic, and algorithmic aspects of generative construction and the development of design as computation via highly coupled parallel explorations of form and description. His past research activities have had a multidisciplinary flavor. He has worked on laser scanning and embedded sensor technologies within dynamically changing construction environments; generative design and sortal representations; object agents in design environments; knowledge-based design systems; integration of natural language and graphics; spatial algorithms; robotic construction simulation; computer graphics and graphical programming environments; and user interfaces for design applications, computer supported collaborative work, and war game simulation. He is currently engaged in research on interior layouts of buildings from their external features, design patterns for parametric modeling, and computational support tools for the design of sustainable buildings.