Pattern Language and Embedded Knowledge in Building Information Modeling

Filiz Ozel
Arizona State University
http:/www.asu.public.edu/~fozel
ozel@asu.edu

When Christopher Alexander (1977), trained both as a mathematician and an architect, published his seminal work "The Pattern Language" in the 1970's and introduced the concept of "pattern language", computers were still in their infancy, CAD did not exist as we know it today, and computer information modeling was not even in the radar screen of researchers. Design communication simply meant manual drafting. With the concept of 'pattern language" (http:// www.patternlanguage.com/), Alexander proposed a systematic method for dealing with complexity, which proved itself to be more relevant than ever in the digital age. The concept is often cited by computer scientists as a precursor to object oriented modeling. This study explores the potential of "pattern language" for structuring building information and design knowledge within the framework of the recent developments in building information modeling (BIM). In this article, comparisons to the approach taken by the software engineering industry who embraced the idea of "patterns" as a systematic way to software development are also made. While Alexander's pattern language proposes a method with which the designer can incorporate his/her experiences and design vision systematically into the process of designing, software industry's approach to patterns describes a method for providing problem and solution patterns (i.e. prototypes) that can be used repeatedly during software development. There is obviously a significant difference between the original intent of the "pattern language" and the way it was later used in other fields including software engineering and business solutions. At the cross section of architectural design and software engineering, Building Information Modeling (BIM) software can benefit from carefully incorporating a combination of these two approaches into its structure as patterns.

Keywords: *Building information modeling; Christopher Alexander; pattern language; software development.*

Introduction

How to digitally model a system as complex and as large as a building has been a concern for researchers and the industry alike for a long time. A model is usually defined as an abstraction of a real world object or phenomenon. This requires the simplification of the phenomenon modeled by selectively including some features while consciously leaving others out. In the past, systems approach has served researchers well, as the key to modeling complex objects during the last four decades has primarily originated from systems theory and systems thinking. This thinking has also been applied to buildings by many researchers where buildings are simply defined as a "set of parts and their connected relationships" (Odum and Odum, 2000).

In modeling of systems, outside influences, component parts, flows and relationships between parts must be considered. Many researchers proposed such frameworks in digitally defining buildings. These range from component models, product models to process models as they pertain to the built environment (Eastman, 1999; Ozel, 2000). Furthermore, there have been numerous efforts to standardize the data modeling of buildings such as STEP (1993), IFC (Industry Foundation Classes) of the IAI – (Industry Alliance for interoperability) etc., led by the architectural/engineering/construction software industry. This has been mostly as a result of concern for interoperability. The key difference between these efforts and "pattern language" is that while the former approaches the issue as an effort to model real world objects in a value free, impartial way, Alexander (1977) intends to include his experiences as a designer into the framework in the form of "patterns" that include rubrics to summarize design rationale. These rubrics as well as the wide range of patterns he proposes clearly allow the embedding of "design knowledge" into design tools, rather than simply approaching the problem as an issue of classification of architectural objects. In short, while pattern language attempts to model knowledge along with information, most object oriented models as well as BIM simply intend to model information in a presumed value and context-free environment. Thus the potential of "pattern language" emerges as it relates to BIM.

While in day-to-day conversation, the words "information" and "knowledge" are sometimes used interchangeably, a careful examination of these concepts reveals considerable differences between the two. While "information" is mostly defined to be factual and value-free (such as the materiality of an object, where they are located in space etc.), knowledge is always based on things learned from past experiences in similar situations and on explicit or implicit assessment of the facts at hand. Therefore a very careful distinction must be made between the two when one attempts to incorporate knowledge as well as information into BIM which often involves the incorporation of domain "knowledge" into the software used.

Patterns

It could easily be argued that Alexander's concept of "patterns" has created more excitement in the software engineering world than it has in the architectural design world. Especially with the advent of programming languages that foster object oriented programming, researchers and practitioners alike started looking for conceptual frameworks that can help the software engineer decide what objects to develop and how to structure them. If for no other reason, for this reason alone, it makes sense to look at the role patterns can play in BIM. On the other hand, software engineering field interpreted patterns in a narrower term than Alexander originally intended. Richard Veryard (2007) asserts that (http://www.users.globalnet.co.uk/~rxv/sebpc/pattcat.htm#purpose) "A popular misunderstanding about patterns is that they are merely abstract components – prefabricated chunks of analysis or design that can be instantly assembled into a solution – the engineering equivalent of convenience foods." He then

continues to emphasize that Alexander's intention with patterns is to lead to a quality design that has "character". In short, rather than being a framework for a generic artifact, pattern use is seen as a framework for unique designs with character. What is most relevant for this study (as we look at how knowledge can be embedded in BIM) is that Veryard (Ibid, 2007) describes pattern language as a "shared repository of knowledge". This is because, both at the object classification level and structural level, Alexander incorporates his own design philosophy, his own world view into "his" pattern language. In a sense, he embeds his own knowledge and understanding of the built environment into the pattern language, thus shows us the way in embedding knowledge and experience into the tools we use for design. This becomes especially important within the context of developing intelligent design tools through digital technology.

The potential of the concept was seen to be so wide spread that it became a catch phrase in a range of fields from software engineering to management and business. Fowler (1995) presented the concept as a "reusability of software modules" issue in his book Analysis Patterns: Reusable Object Models., as did Gamma et al. (1995) in their book titled "Design Patterns: Elements of Reusable Software". On the other hand, the business world started using the term "pattern" to indicate "best practices", although "pattern" indicated a more systematic approach to the concept of "best practices". While the latter implied a random collection of approaches to solving business problems, the former was seen to be more systematic and more helpful as an analytical tool, as in Adrian Slywotzky et al 's (1999) book titled "Profit Patterns" and in IBM's approach to "e-commerce patterns" (http://www-128.ibm.com/developerworks/patterns/).

IBM defines business patterns as "the interaction between users, businesses, and data. Business patterns are used to create simple, end-to-end e-business applications." They continue to define e-commerce development issues by saying "Composite patterns are combinations of Business patterns and Integration patterns that have themselves become commonly used types of e-business applications….." In a sense, IBM is describing prototypical e-commerce applications in what they call "composite patterns", using the term pattern to be synonymous with prototype. It is in the former sense, i.e. in its potential to provide a systematic framework for analysis and synthesis that can lead to design, this researcher finds the concept of "patterns" to be most useful for architectural design and for BIM, while "pattern" as a prototypical solution to a given design problem is a much narrower interpretation of the concept that might have some potential for BIM. The concept of "model driven software development" also implies prototype as a starting point for software design (http://www-128.ibm.com/developerworks/rational/products/patternsolutions/). IBM further asserts that "pattern implementation, is assisted by tooling capabilities such as those found in Rational Software Architect and enables patterns to be easily shared and applied." This in a sense brings the issue of smart tools to the forefront of the discussion, and can lead in the direction of incorporating prototypical solutions into software for further revision and adaptation.

As a summary, the concept of patterns in design has been interpreted in two different ways by the software design and business community, one as an analytical tool that allows the incorporation of one's own experience and worldview into the "patterns" of design, the other as a method to develop reusable prototypes that reduce design and production time and error. The use of patterns in BIM probably lies somewhere in between. While, some architectural design problems are more amenable to prototypical solutions, most are not. On the other hand, unique architectural design approach of a particular architect or design firm can be analyzed as a set of patterns of design, thus leading to unique designs. In either case, the concept of "pattern" can help systemize the process of design, while the final artifact remains to be a unique solution to a given design problem. In a

sense, design problems and processes can be analyzed to indicate a pattern, but that does not mean the solution must also abide by a pattern (i.e. by a prototypical solution). Therefore, the concept will be further investigated within the framework of "process" rather than within the framework of "product" or "prototypical solutions".

Patterns for the built environment – Christopher Alexander

Alexander's patterns for the built environment cover a very wide range of scales, from the level of the region to the level of the individual room and its ornaments (as he calls them) (Alexander et al., 1977). He somewhat extends the patterns to the construction stage as well. Patterns in this framework are much more than objects listed in a hierarchic manner, although many of them are what can be called objects (http://www.patternlanguage.com/apl/aplsample/aplsample.htm) in object oriented programming sense as well as in architectural sense. In the sampler posted at this WEB site, 253 patterns are listed for the design of a house as does his book published in 1977. These range from a region to an individual room in a house. Although these are listed in a hierarchic manner based on physical scale, the following types of elements are listed as part of the patterns:

1. Rationale – Each section starts with a narrative that summarizes the overarching principle (a rubric) that guides the design of the built environment at the selected level immediately below. For example, for regions, the overarching principle is listed as "Within each region work toward those regional policies which will protect the land and mark the limits of the cities". Whereas for the skin of a building, the following rubric is listed: "Prepare to knit the inside of the building to the outside, by treating the edge between the two as a place in its own right, and making human details there." For site planning, the following design principle is brought forth "Between the house clusters, around the centers, and especially in the boundaries between neighborhoods, encourage the formation of work communities." Therefore design rationale is an important part of the patterns listed. Rationale represents the knowledge and experience of the designer, thus reflects the knowledge embedded in the pattern framework.

2. Object patterns – These can be grouped as:

a. Physical object patterns for objects such as "main entrance", "half hidden garden", "entrance transition", "roof garden". These objects are rarely listed without a qualifier, thus include the designer's values and judgment calls. These in the object oriented sense, are objects that can be defined as entities with properties and behaviors. These "desired (by the designer)" properties and behaviors represent the knowledge and experience of the designer embedded in the pattern language.

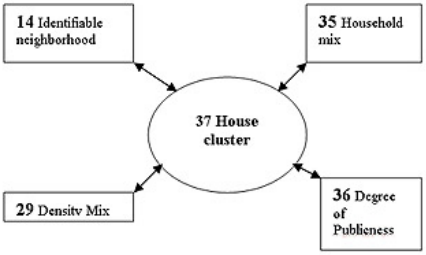
b. Geometric/formal patterns – Alexander also specifies patterns of formal organizations with specifications that include the expected performance from the proposed geometric/formal pattern. An example of this is regarding housing clusters. He specifies to "arrange houses to form very rough, but identifiable clusters of 8 to 12 households around some common land and paths" (http://www.patternlanguage.com/).

c. Social Patterns – Social activities that represent a design paradigm (i.e. have an impact on design) are included as social patterns. Among these are patterns such as "dancing in the street", "connected play", "small services without red tape". Many of these come with qualifiers. These social patterns are often networked with physical object patterns as part of the relationship network.

3. Qualitative Patterns - These are simply qualifiers listed independent of a particular object but can be applied globally to the objects designed. Examples of these are: "wings of light", 'activity pockets", "something roughly in the middle", "indoor sunlight", "tapestry of dark and light", "connection to the earth" etc.

4. Relationship patterns – Alexander (1977) builds a very deliberate network of relationships between all patterns. These are labeled as "relationship patterns" in this article. They can be either between

Figure 1
An example of a pattern for a relationship network



two physical objects or between a physical object and an activity, the latter being more common. Examples of the latter are: "windows overlooking life", "common areas at the heart". Whereas the former are: "private terrace on the street", "sequence of sitting spaces", "closets between rooms" etc. He uses a numbering system to build a network representation of relationship patterns, each individual pattern cross referenced by its number (fig. 1).

Software design pattern solutions

Software industry mostly utilizes the concept of pattern in specifying software solutions to typical business problems. As part of a model-driven software development (MDSD) environment, IBM corporation provides and builds customized business applications by making use of the concept of pattern extensively. IBM indicates that patterns "…assist in the creation of artifacts according to best practices through all stages of the solution development life cycle." (http://www-128.ibm.com/developerworks/rational/products/patternsolutions/). A formal description called pattern specification captures the knowledge about a problem and its solution. IBM defines a pattern specification as "… a precise definition and description of a pattern in conceptual terms that allows for its communication and reuse by others." Pattern specifications are comprised of:

- Context: The rationale for applying the pattern (parallels can be drawn to Alexander's formulation of a design rationale)
- Problem: Precise statement of the problem the pattern solves (Ibid, 2007)
- Solution: Description of the solution the pattern provides (Ibid, 2007)
- Results: Consequences and advantages and disadvantages of applying the pattern (Ibid, 2007)

Therefore, while Christopher Alexander's pattern language is not problem centric, IBM's is. Furthermore, the former loosely specifies partial solutions (each called a pattern) to a hierarchy of design problems, while IBM labels a combination of a well defined problem and its best practice solution a "pattern". Alexander does not claim to be summarizing best practices, although there is an implication that designs that follow his patterns are "good" designs. Therefore, in a sense they are presented by Alexander as "his own set of best practices (and best qualities of design)".

Patterns in Building Information Modeling

The A/E/C software industry primarily defines BIM as information modeling an example of which can be found at Autodesk's BIM Website: "An Autodesk innovation, building information modeling (BIM) is the creation and use of coordinated, consistent, computable information about a building project in design – information used for design decision making, production of high-quality construction documents, predicting performance, cost-estimating and construction planning, and, eventually, for managing and operating the facility." http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=8127972. This definition misses acknowledging the role knowledge plays in A/E/C decision making. On the other hand, the structure of information in BIM and the selection of the entities that are included in BIM cannot be done without knowledge regarding the way A/E/C processes work. Graphisoft's Virtual Building concept in its ArchiCAD software (http://www.graphisoft.com/products/virtual_building/) is a good

example of this where the 3D model of the elements and the 3-D database must rely on knowledge about how buildings are put together. This also applies to Autodesk's software as well as to other BIM software.

One can easily argue that AEC software firms use the concept of "patterns" in their software (although they do not label them as such), since they seem to work with preconceived ideas about what the design process and production is about and which typical "design process patterns" and "design object patterns" are to be incorporated into their software. This gained importance especially as the industry transitioned from a worldview of drafting-support (i.e. CAD) to a world-view of design-process-support; i.e. the industry had to base their object models on real world design activities (as they see and analyze them).

One other source of possible patterns in BIM software comes from the user, i.e. from the architect, the designer and/or the design firm. It is critical to make this differentiation, as one looks at "patterns" that can be built into BIM software. The word "pattern" is used here in the sense Alexander is using and as Veryard has very eloquently put it: they are a method of embedding knowledge into a process (whether it is electronically supported or not). Thus the source of BIM patterns (i.e. embedded knowledge) can come from:

- The BIM software designer
- The BIM software user (the designer or the design firm).

Therefore, extensibility is an important aspect of BIM software just as it was an important aspect of CAD software. Table 1 summarizes and compares patterns incorporated into four different domains discussed in this article, namely, Alexander's patterns, software engineering patterns in general, BIM software designer's patterns and architect's/design firms patterns created and embedded into BIM software. There are obviously parallels as well as disjunctions between the characteristics of these sets of patterns.

Summary

A close inspection of the twelve types of patterns listed in Table 1 indicates that many of these are missing in current BIM software (as originally designed by the AEC software builder), but through customization by the designer and/or the design firm, many of the pattern-types can (and are) incorporated into BIM applications. Among the types from Alexander's patterns that are most often missing in customized BIM use (i.e. the rightmost column in Table 1) is design rationale, qualitative patterns (i.e. desired performance indicators) and social patterns. Others are often incorporated at varying levels. An important additional one that is missing is what this researcher calls relationship patterns. This is defined as a network of relationships between different types of patterns. Any relationship between objects that exists in BIM is intended as value-free definitions of the physical aspects of a building. Whereas, Alexander uses relationship patterns to describe his vision for the built environment, thus includes qualitative patterns (e.g. light from both sides), objects with qualifiers (e.g. identifiable neighborhood), social patterns and others in the relationship patterns he builds (see figure 1 as an example, but also see http://www.patternlanguage.com/ for a long list of relationship patterns identified for every numbered pattern Alexander lists.) Regardless of the content of each pattern, to the degree that an architect makes a conscious effort to identify relationship patterns in BIM, he/she can incorporate his/her own vision into the design process using the tool. Therefore, what is argued here is not to adopt the patterns as exactly identified by Alexander, but to adapt his approach to pattern definition as a method for embedding design knowledge and past experience into BIM.

Table 1
A comparison of the characteristics and structure of models of four different pattern paradigms

Characteristics of Pattern models	Alexander's patterns	Software engineering patterns (IBM and others)	Software designer's BIM patterns	Architect's/Design Firm's patterns for BIM
Rationale part of a pattern	Yes	Yes (called "context" by IBM)	No	Maybe (depends on how the design firm has set up BIM in their office)
Physical patterns (design objects)	Yes	No (not applicable)	Yes	Yes
Geometric patterns	Yes	No (not applicable)	Yes (at architectural element level)	Yes
Social patterns	Yes	Sometimes (as part of business patterns)	No	Sometimes
Business patterns	No	Yes	Design business patterns	Design business patterns
Design Communication patterns	No	Yes	Yes, a major component of the software (e.g. Layers, layout design, etc.)	Yes, customization based on design firm's established visual language
Relationship patterns	Yes (explicitly defined and built through a network of patterns)	Yes (implied- only between objects, but not between other patterns)	Yes (implied- only between objects, but not between other patterns)	Yes (implied- only between objects, but not between other patterns.)
Qualitative patterns (overarching qualities)	Yes (designer's own judgment as well as use of evidence from research)	No	No	No
Problem patterns	Partially exists, but not the main intention	Yes, an important aspect of the approach	Somewhat (but mostly as design production patterns rather than at the conceptual level)	Sometimes, especially if the design firm specializes in the design a particular building typology
Solution patterns (prototypical solutions)	Suggested partial solution patterns are given	Yes Solution patterns are systematically developed and made available to the software designer	Very limited. Generative patterns (such as that for a spiral staircase) can be classified as such.	Many times yes. Partial solution patterns are often part of design library of a design firm
Result patterns (analysis of a given solution pattern)	No (although some analysis of the projected performance of a given object is provided as part of the rationale)	Yes (Consideration of the results is a requirement of the process used)	No, unless analytical software such as that for energy analysis are added as a module	No, unless analytical software such as that for energy analysis are added as a module
Production patterns (construction)	No (very limited and very broad in scope)	Yes (part of software structure and solution patterns)	Yes	Yes

References

Alexander, C., Notes on the Synthesis of Form, Harvard University Press, 1964.

Alexander, C., The Timeless Way of Building, Oxford University Press, 1979.

Alexander, Christopher et al., A Pattern language, Oxford University Press, 1977.

Autodesk building solutions, Building Information Modeling in Practice, http://images.autodesk.com/adsk/files/bim_in_practice.pdf.

Autodesk Inc., Building Information Modeling for Sustainable Design, http://images.autodesk.com/adsk/files/bim_for_sustainable_design_jun05.pdf.

Autodesk http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=8127972, April 2007.

Eastman, C., Building Product Models: Computer Environments Supporting Design and Construction, CRC Press, Boca Raton FL, 1999.

Fowler, Martin, Analysis Patterns: Reusable Object Models, Addison-Wesley Longman, 1995.

Gamma, Erich, Helm, Richard, Johnson, Ralph & Vlissides, John. Design Patterns: Elements of Reusable Software. Addison-Wesley Longman 1995.

Graphisoft Inc., http://www.graphisoft.com/products/virtual_building/) Virtual Modeling Software, ArchiCAD, .

http://www.livingneighborhoods.org/ht-0/gcwelcome.htm.

IBM Business solutions WEB site, http://www-128.ibm.com/developerworks/patterns/.

International Alliance for Interoperability: 1998, Industry Alliance for Interoperability; http://www.iai-na.org/.

Odum, Howard T. & Odum, Elisabeth, C., Modeling for all Scales, Academic Press, San Diego, CA, 2000.

Ozel, F., "Spatial Databases and the Modeling of Dynamic Processes in Buildings", article published in the Proceedings of the CAADRIA 2000 conference, Singapore, May 2000.

Pattern Language, http://www.patternlanguage.com/.

Simon, Herbert, Models of Man: Social and Rational, John Wiley and Sons Inc., New York, NY, 1957.

Slywotzky, Adrian et al., Profit Patterns, US edition: Random House, 1999. UK edition: John Wiley, 1999.

STEP: "Standard for the Exchange of Product Data", ISO CD 10303, 1993.

Ullman, Jeffrey D., Principles of Database and Knowledge-base Systems, Computer Science Press, Maryland, 1988.

Veryard, Richard, (http://www.users.globalnet.co.uk/~rxv/sebpc/pattcat.htm#purpose), May, 2007.

Winograd, T., & F. Flores, Understanding Computers and Cognition: A New Foundation for Design, Addison-Wesley, 1986.