A Form-making Algorithm

Shape Grammar Reversed

Park, Hyoung-June and Emmanuel-George Vakaló
University of Michigan

Key words: CAAD, Design Research, Design Languages, Design Process Model, and Internet

Abstract: Under the assumption that design is a hypothetical test of building, this paper introduces a method of employing architectural knowledge through direct manipulation of geometric objects. Proposing a framework for retrieving and analysing not only what is modelled but also how it is designed, this paper demonstrates that ***designing*** can be viewed as an object of research. The paper also discusses the issues pertaining to the implementation of the aforementioned framework.

1. INTRODUCTION

Based upon the possibilities of exploring spatial design with Froebel's gifts, Stiny, in his article "Kindergarten Grammar," (Stiny, 1980) develops a visual grammar to formalise a vocabulary of building elements and a system of categories of forms in languages of designs. The languages are formed by combining or augmenting other languages of designs in terms of various language-theoretic operations such as substitution (addition and deletion,) Boolean operations (union, intersection, difference) and basic transformation functions (translation, rotation, mirroring, and scaling.) Stiny shows possible techniques of identifying a spatial relation in his grammar.

After this, a number of efforts have been made in making an application of a three-dimensional shape grammar. Piazzalunga and Fitzhorn sketch a possible way of recognising a three-dimensional shape grammar, and a framework of the shape grammar application (Piazzalunga and Fitzhorn, 1998). Agarwal, Cagan, and Constantine suggest the idea of optimising production system according to the feedback about each separate stage of designing (Agarwal, Cagan, et al., 1998). Nevertheless, the use of shape grammar still has been difficult for a student and a designer who are more familiar with manipulating formal objects than with making a rule and applying the rule to design. It is because a rule application algorithm in shape grammar generates a low degree of freedom in design with imposing a generating a rule itself as a constraint of design.

With the review of the current rule application algorithm in shape grammar, this paper introduces an algorithm of form making based upon designer's direct manipulation of objects as a possible solution to the rule-generating problem embedded in current shape grammar application. According to this form-making algorithm, a framework of computational application is proposed in this paper.

The framework has three theoretical buttresses for its implementation. The first buttress entails ***formalisation*** that translates a designer's form-making process into a set of "form-making rules."[1] The second involves ***data abstraction*** that stores the design algorithm, which is generated during design process, into a database in the format of the "Extensible Mark-up Language (XML)" (Maruyama, Tamura, et al., 1999). This form-making information is stored when a designer derives a new design object with spatial relations between different objects. During this storing process, "Object-Relational Database Management System (ORDBMS)" (Soutou, 2000) is employed for maximising the ability of querying and accessing design knowledge in database. The design knowledge consists of the annotations of each design object and the relation between different design objects. The third entails ***communication*** that applies each stored information to design. The stored information inside the database is exchanged throughout the Internet.

The framework proposed in the paper allows a designer to manage each design object in three different types of information such as "*Artifact*,"[2] "*Building Information*,"[3] and "*Design Algorithm*."[4] Three different types of design information embedded in a design object allow the designer to analyse and modify various aspects of his or her design. Now, an individual way of designing, tacit knowledge, becomes an object of play.

[1] The form-making rules consist of basic transformation functions (translation, rotation, mirroring, and scaling), spatial relations, which are addition, deletion, and Boolean operations (union, difference, and intersection), and spatial elements such as points, lines, planes, and solids.

[2] A visualised geometric representation of a design object

[3] An architecturally categorised information of constructing a design object

[4] A set of form-making rules (schemas) established by a designer's direct manipulation of a design object

2. A SHAPE GRAMMAR

As defined by Stiny, a shape grammar is a four-tuple (S, L, R, I), in which (1) S is a finite set of shapes; (2) L is a finite set of symbols; (3) R is a finite set of shape rules; (4) I is an initial shape. Design solutions defined by a shape grammar are generated by applying the shape rules in the set R to the initial shape I and to shapes produced from I. In specific, the set of sequentially ordered rules for making a design solution is "schema." A shape rule in the set R has a normal form **A → B**, where **A** and **B** are labelled shapes in $(S, L)^+$ [5]and $(S, L)^*$ [6], respectively. In this paper, all shapes are regarded as solids in space, which are shapes in U_{33}. Stiny defines the algorithm of shape rule application in his article "Shape Rules: closure, continuity, and emergence" (Stiny, 1994) as follows;

If you have a rule for your design developing	**A → B**
Apply your rule to a shape	**C**
if a transformation of **A** is a subset of **C**	$\mathbf{T(A) \subseteq C}$
Delete a transformation of **A** from **C**, and add a transformation of **B**	$\mathbf{(C - T(A)) + T(B)}$
Then, you will have a new shape **C'** form **C** with a rule **A → B**	**C'**

This algorithm has been employed as a method for analysing architectural precedents. Several grammars such as "Palladian Grammar (Stiny and Mitchell, 1978)," "The Grammar of Paradise (Stiny and Mitchell, 1981), " "The Language of Prairie: Frank Lloyd Wright's Prairie Houses (Koning and Eizenberg, 1981)" showed the possibility of using shape grammar in the research of traditional buildings. However, the burden of generating a rule to apply the rule itself to design has limited the usage of shape grammar in design practice.

Without a certain rule or intention of developing a design process, the current algorithm of shape grammar cannot proceed any further design step as illustrated above. In addition, another problematic point is that most designers do not have a rule or solution to every design problem. It leads a designer to manipulate shapes for finding or generating a rule. The current algorithm does not sufficiently explain a connection between designer's manipulation and generating a rule in shape grammar. In addition, the current algorithm does not clearly explain how to introduce the interpretation of the semantic part of architectural design although it effectively represents the syntax of the design. Comparing to shape grammar explained by Stiny, Chomsky's generative grammar (Chomsky, 1978) consists of the basic

[5] $(S,L)^+$ is a set which contains all labelled shapes made up of shapes and symbols in the set S and L

[6] $(S, L)^*$ is a set which contains $(S,L)^+$ and the empty labelled shape $<S_\varnothing , \varnothing>$

components, which are lexicon and rewriting rules. Lexicon represents a list of ***words***. It shows that the generative grammar concerns a meaningful language, which is composed of ***words***. However, Stiny's shape grammar represents a world with geometric elements, which may be compared to ***letters or alphabets*** instead of ***words*** that have meanings. Therefore, without an introduction of meaning embedded in design object, a shape grammar may produce an ambiguity in terms of confusion not creativity.

3. A FORM-MAKING ALGORITHM

The importance of direct manipulation of object in design process has been acknowledged among architects and designers since Frank Lloyd Wright stated, in his biography, the influence of playing Froebel's gifts in kindergarten method on his design. Also, it is highlighted by a few American pragmatists such as Pierce and Dewey. Dewey describes a pattern of design action in inquiry as "the controlled or directed transformation" (Dewey, 1986). Pierce suggests that the habits of purposeful actions are the rules or patterns of solving problems in the process of inquiry (Pierce, 1966).

With the hypothesis that direct manipulation of object can be the rules of making a transient progress of design or inquiry, an algorithm for translating the manipulation into a rule during the design process is proposed below.

```
When you have a shape $\mathbf{A}$ to be developed                                   $\mathbf{A}$
Make a transformation of a shape $\alpha$                                          $\mathbf{T}(\alpha)$
such that $\alpha \in \{ \varnothing, \ldots, \mathbf{A}, \ldots, * \}$
Then, define a spatial relation $\otimes$ between $\mathbf{A}$ and $\mathbf{T}(\alpha)$        $\mathbf{A} \otimes \mathbf{T}(\alpha)$
such that $\otimes \in$ { addition, deletion, union, difference, intersection}
Whenever you get $\mathbf{B}$ such that $\mathbf{B} = \mathbf{A} \otimes \mathbf{T}(\alpha)$
The relation between A and B is stored as a rule                                   $\mathbf{A} \rightarrow \mathbf{B}$
```

Where $\mathbf{A}$, $\mathbf{B}$, and α are in U_{33} (Solids in Space) Also, the addition is $\varnothing + \alpha$ and the deletion is $\beta - \alpha = \varnothing$ such that $\varnothing$ is empty shape and $\beta \subseteq \alpha$ where α and β in U_{33}.

With the suggested algorithm, a designer can focus on his/her design without the burden of shape rule making. Whenever a design solution is achieved through substitution or Boolean operation between an initial shape and a transformed shape, the relation between the initial shape and the solution is stored as a rule in a machine. At each cycle of this algorithm, a designer is able to attach an architectural meaning to each solution. This process provides a mapping of designer's meaning to a rule in shape grammar. The mapping leads a designer to apply the stored rules for solving other design problems. At the end of design process, the whole series of shape rules of a final design object are organised with proper meaning attached in a machine. This architectural reference mapped to a final design object allows a designer to change the part of his/her final design result not only modifying a shape/ design object but also alternating the rule assigned to the shape/ design object. Thus, a syntactic intervention of design process is possibly achieved with modifying the rules generated by designer's direct manipulation of shapes during design process.

4. MAKING AN ARCHITECTURAL REFERENCE

Based upon the proposed algorithm, a tool for making an architectural reference is introduced. The tool employs the notion of "*object*"[7] and suggests a way of understanding design as a process of making "*a meaningful order*" (Papanek, 1984) and a set of building information annotated to a designed shape in U_{33}, a solid in space.

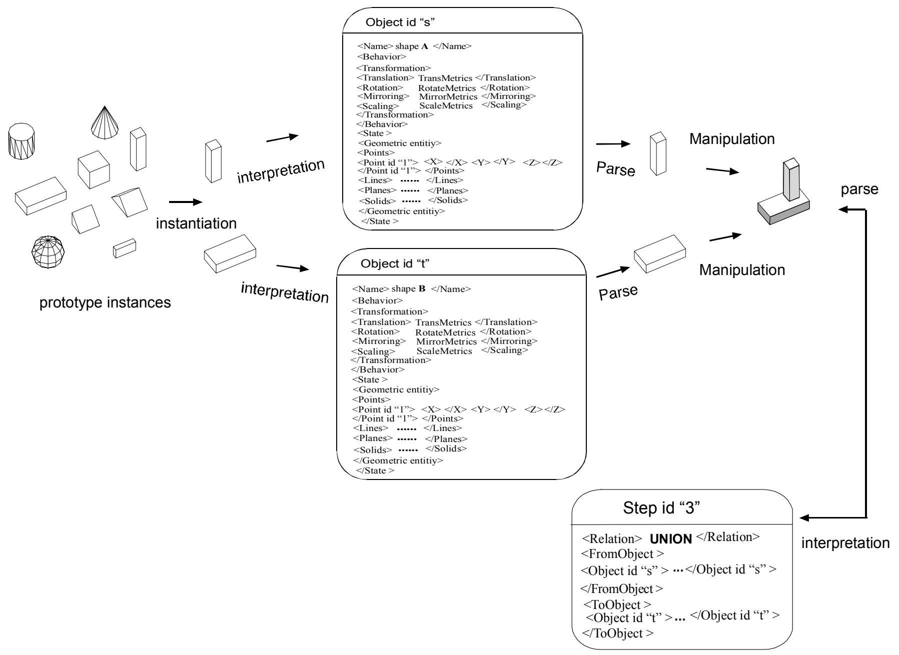
The proposed tool regards all the components used to generate a design as a set of *object*s, which are organised in sequence of design resolution. The basic structure of *object* consists of state and behaviour. State contains geometric entities as attributes of *object*. Behaviour has basic transformation functions. In addition, the spatial relation between different shapes in U_{33} defines a step, which generates a new shape. The relation includes addition, deletion, and Boolean Operations. A step *object*, which is the design resolution, clarifies schema known as series of rules. Then, each step *object* is embedded as one of attributes of the building information of a new shape. The new shape is represented in two different aspects. The first aspect is the shape as an *object* containing geometric information. The second is as an *object* in the spatial relation with other *objects*. Therefore, with this architectural reference, a user of this tool will get geometric data of a designed shape and his/her design algorithm of deriving the shape.

4.1 Formalisation

Formalisation allows a user of this tool to define shapes in U_{33} as *objects*. When the user makes addition or deletion of a shape with instantiating prototypes or using previously defined the shape, the tool creates an *object*. The attributes of the *object*, which are geometric entities, are established either by the user or defined as default value initially by the tool. Either the tool or the designer gives corresponding label or name to the *object*.

[7] An object always has two characteristics: state and behaviour. For example, Bicycles have state (current gear, current pedal cadence, two wheels, number of gears) and behaviour (braking, accelerating, slowing down, changing gears)

However, only the tool defines the identity number of the *object*. In addition, the behaviour (basic transformation functions) of the *object* is defined by the user's direct manipulation of the shape and organised by the tool. Also, the user creates a new shape by making a spatial relation between different shapes. This process is recorded as a step *object*. The created *objects* are also stored as instances for the future usage.



```
<Name> shape A </Name>
<Behavior>
<Transformation>
<Translation> TransMetrics </Translation>
<Rotation>    RotateMetrics </Rotation>
<Mirroring>   MirrorMetrics </Mirroring>
<Scaling>     ScaleMetrics  </Scaling>
</Transformation>
</Behavior>
<State >
<Geometric entitiy>
<Points>
<Point id "1">  <X> </X> <Y> </Y>  <Z></Z>
</Point id "1"> </Points>
<Lines>  ......  </Lines>
<Planes> ......  </Planes>
<Solids> ......  </Solids>
</Geometric entitiy>
</State >
```

```
<Name> shape B </Name>
<Behavior>
<Transformation>
<Translation> TransMetrics </Translation>
<Rotation>    RotateMetrics </Rotation>
<Mirroring>   MirrorMetrics </Mirroring>
<Scaling>     ScaleMetrics  </Scaling>
</Transformation>
</Behavior>
<State >
<Geometric entitiy>
<Points>
<Point id "1">  <X> </X> <Y> </Y>  <Z></Z>
</Point id "1"> </Points>
<Lines>  ......  </Lines>
<Planes> ......  </Planes>
<Solids> ......  </Solids>
</Geometric entitiy>
</State >
```

```
<Relation> UNION </Relation>
<FromObject >
<Object id "s" > ...</Object id "s" >
</FromObject >
<ToObject >
<Object id "t" > ... </Object id "t" >
</ToObject >
```

Figure 1. Formalisation

4.2 Data Abstraction

Data abstraction enables a user of the tool to isolate "how a compound *object* is used from the details of how it is constructed from more primitive *objects*. " (Abelson and Sussman, 1996) Assuming the tool provides the basic transformation functions of spatial entities, the instantiated *objects* can be assembled with various sets of formal relations defined by the user. The tool organizes each step of creating another/new *object* according to the user's assembling of shapes in U_{33}. Illustrating steps of the user's design process as schema, the tool provides an understanding of the evolution of a shape and a set of rules that generates the shape. Therefore, the sequence of making a new shape is displayed as a step *object*, and the transformation of a shape is explained as an *object* in XML format. With the structured design information, the user of this tool is able to assemble the subset of the shape not only in constructing a compound shape within a visual state but also in syntactically modifying the process and transformations of the *object*. Figure 2 shows the data abstraction layers of the *object*, step *object*, and schema.

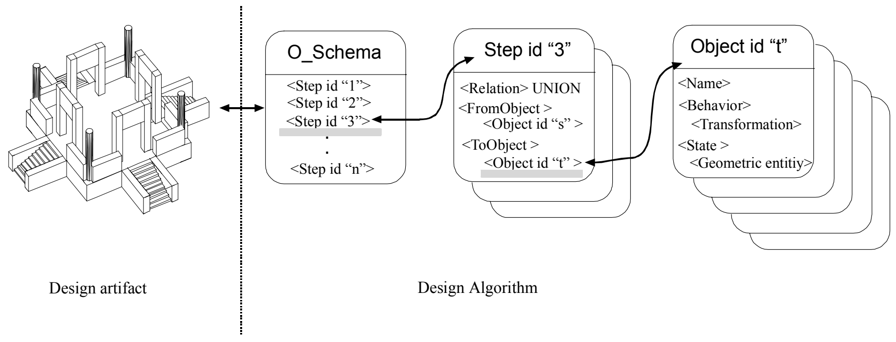


Figure 2. Data Abstraction Layers

XML is a way of structuring information in cross platform. Also, XML is syntax for establishing the formation of the hierarchical containers, which include various data type. Therefore, basically an XML document is a tree of elements in a certain order. Since XML allows a user to define the structure of each document within document-type definition (DTD), it has the advantages of describing meta-content, publishing database contents, and communicating data using a messaging format. With these advantages, XML provides a possible way to retrieve a design information as the set of programming codes instead of the format of DXF (Data eXchange File.) Thus, it helps reducing the size of memory for saving a design information. Also, XML allows a user to understand design/designing as a procedural development of structured information with manipulating shapes. The structured information is managed in ORDBMS. ORDBMS is employed for managing the storage of an organised design information. As an extension of Relation Database system (RDBMS) for affording Object-Oriented design concept, ORDBMS provides user-defined types including data structures, collection, encapsulation, inheritance, and Object Identity. Collections are a means of storing a series of data entries as a group. Encapsulation implies that data abstraction and data hiding. It also provides the hierarchy between data objects. Inheritance implies that a child of father data object can have the characters of the father object. With these features, ORDBMS allows the designer to perform complex analytical and data manipulation for searching and retrieving various objects. Comparing to Object-Oriented Database System, ORDBMS provides frequent querying / updating access to large collection of data. (Ramakrishnan, 1998) In this proposed tool, this factor is vital since a design information is supposed to be generated whenever a new shape is created from a designer's manipulation. Figure 3 illustrates the data model of each information object managed in Oracle Designer 6.

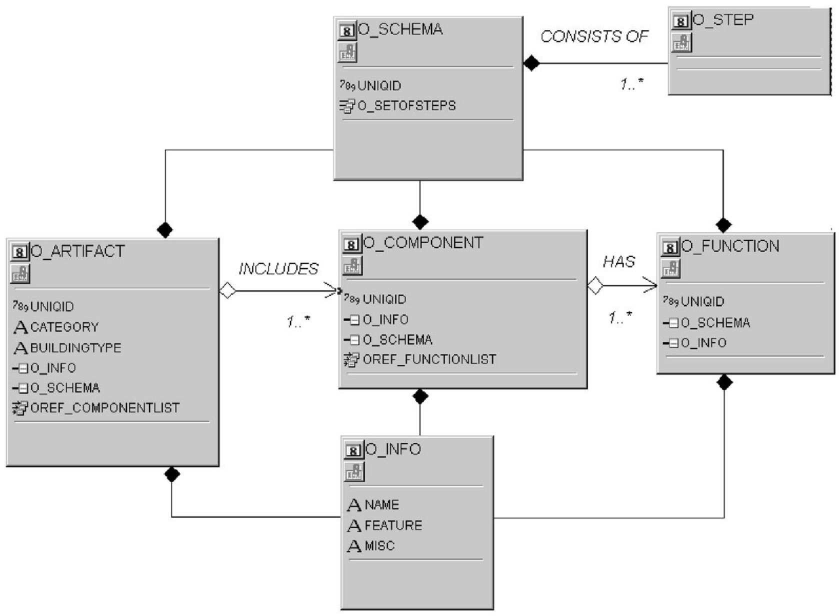


Figure 3. Data Model

4.3 Communication

To apply the proposed architectural reference within ORDBMS, establishing a network of communication is critical. It is necessary for facilitating, through the web-browser, the exchange of encapsulated information in the database with other architectural references. With the exchange, feedback and error-elimination are requested to the user of this proposed tool. According to the result of communication, the user can change the state and composition of shapes by altering the *object*, step *object* and building information parsed inside the database.

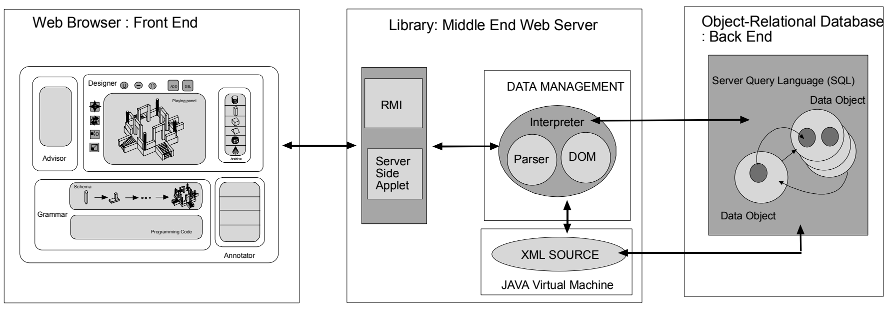


Figure 4. Communication Network

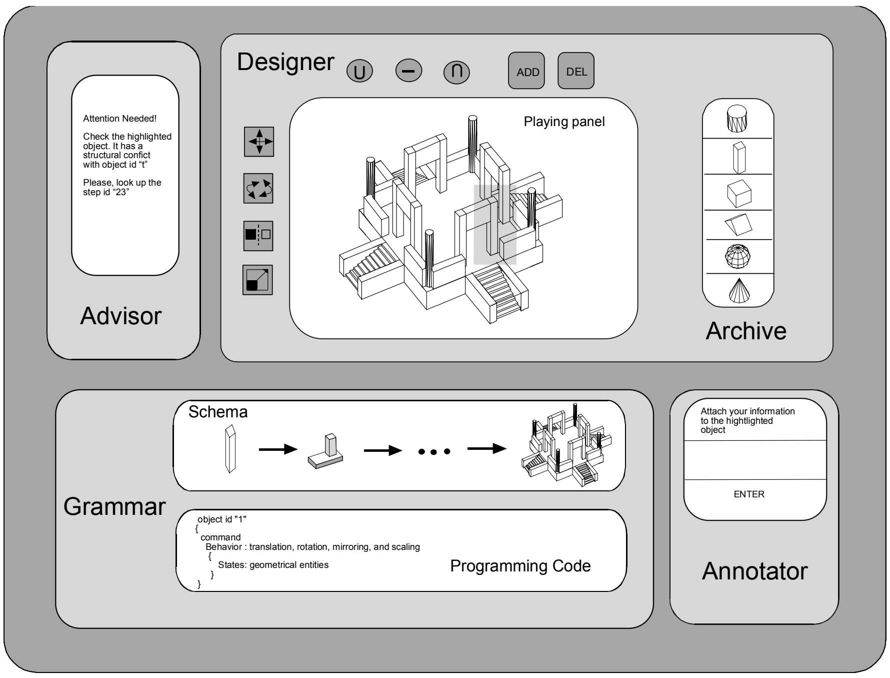
Initially **Archive** in the front end contains sets of prototype shapes as *objects*. The sets of prototype instances are pre-defined. However, in **Archive,** the number of *objects* and step *objects* increases when a new shape is constructed in **Designer** interface. According to changes occurred either in **Designer** or in **Grammar, Interpreter** in the middle end translates the shape constructed by the user into *object* and step *object* in the format of XML. In return, **Interpreter** illustrates the parsed *object* on the **Archive** panel in **Designer**, and design developing procedures on the **Schema** panel in **Grammar**. The components of the tool are illustrated in Figure 5.

The interpreted shape in XML is the subset of an *object* that makes up a final shape. It leads the user to find out how his or her final shape consists of sets of individual shapes. Also, **Interpreter** in the middle end converts each command of search conducted in **Annotator** to Server Query Language (SQL) for proper function of ORDBMS. In addition, **Interpreter** re-organises the parsed step *object* in LSP format as a programming code. In return, the programming code itself is appeared on the code panel in **Grammar**. The final shape is divided into *object*s and step *object*s in the format of XML during a user's design developing. With the help of network communication based upon "Remote Method Invocation (RMI) and Server Side Applet (SERVLET)," (Reese, 1997) the user is able to employ the basic transformation functions and spatial relations, which are transferred to the database in the back end server, in **Designer** or **Grammar**.

4.4 Implementation

4.4.1 Components

The components of the tool for making an architectural reference are **Designer**, **Grammar**, **Annotator**, **Archive**, and **Advisor**.



Attention Needed!

Check the highlighted object. It has a structural confict with object id “t”

Please, look up the step id “23”

```
object id "1"
{
  command
    Behavior : translation, rotation, mirroring, and scaling
     {
        States: geometrical entities
      }
}
```

Attach your information to the hightlighted object

Figure 5. Use-Interface

Designer gives a user of this tool several options for developing his/her design such as basic transformations, substitution, and Boolean operations. Also, **Designer** visualises design shapes in U_{33} according to the user's direct manipulation of the shapes. In **Grammar**, the user is able to perform a syntactical intervention of design process by modifying a design algorithm, which is generated from **Designer**. The intervention is made with changing contents in **Programming Code** or altering the sequence of the evolution of shapes in **Schema**. Also, the intervention made in **Grammar** effects the state of a shape represented in **Designer**. With **Annotator**, the user can attach a building information to each shape according to his/her need. In addition, **Annotator** helps the user search and update the instances in **Archive**. The search and update method leads the user to make a systematic comparison. During design process, the category, type, function and constructional information of each shape are organised as the contents of a data object named "O_Artifact" in a database. It allows user to study his/her design with adequate information. In **Advisor** panel, possible problems embedded in each shape are displayed based upon the exchange of design information among different architectural references.

4.4.2 From Shape to Object

When a user selects and transforms a certain shape, a temporary memory is cached for storing the state of the selected shape. Comparing the state and the changed state, the transformation used for the changes is specified: A = B * T, then B^{-1} * A = T where A is a 4 x 4 matrix of the selected shape. B is a 4 x 4 matrix of an altered shape. B^{-1} is an inverse matrix of B. T is the basic transformation function matrix (Translation, Rotation, Mirroring, or Scaling). The initial states of shapes are already known since the user is supposed to begin his/her design with instantiating prototypes contained in **Archive** panel of **Designer,** which is in front end. According to information of the states and behavior of a shape, **Interpreter** in the middle end writes an *object* as a description of the shape in XML format.

4.4.3 From Object to Step Object

A step *object* is specified only when a user makes a new shape by defining a spatial relation between shapes. The spatial relation is defined by the user's direct manipulation of the shapes. And the identity number of step *object* is automatically defined in sequential order only by ORDBMS. The step *object* contains FromObjects, ToObjects, and the spatial relation. Especially, ToObject of addition is always Ø (void)

4.4.4 From Step Object to Programming Code

```
<Step id ="3">
   <Relation>add/delete,Boolean operation </Relation>
      < FromObject >
         <Object id ="1">
            <Name></Name>
            <Behavior> Transformation</Behavior>
             <State>geometrical entities</State>
         </Object>
      </ FromObject >
      <ToObject >
         <Object id ="2">
            <Name></Name>
            <Behavior> Transformation</Behavior>
             <State>geometrical entities</State>
         </Object>
      </ ToObject >
</Step>
```

```
{Step3
   command
  {
     Retaion : addition/deletion/ Boolean Operation
        {
          FromObject
          {
            command
            {
              object id "1"
              {
               command
                 Behavior : translation, rotation, mirroring, and scaling
                 {
                    States: geometrical entities
                  }
              }
            }

          ToObject
          {
               command
               {
                 object id "2"
                {
                 command
                   Behavior : translation, rotation, mirroring, and scaling
                   {
                      States: geometrical entities
                   }
                 }
               }
            }
          }
     }
}
```

5. CONCLUSION

The tool for making an architectural reference has been developed as a migration of Nine Square Grid Composition (NSGC), which was designed within AutoCAD environment in AutoLisp and DCL, to a web-based application in C++ and JAVA programming language. The basic structure of NSGC is rooted from research on designer work in traditional studio class.[8] There are two most polemic points taken from the research. The first is that rational discussion between designer and instructor about a design is not possible without records of the form-making process. The second is that the necessity of an architectural reference for searching, comparing, and retrieving design artifacts during design process with a computational application.

With the proposed form-making algorithm focused on a direct manipulation of design objects and its translation to a rule, a tool for making an architectural reference shows the possibility of implementing shape grammar in constructive design practice. By introducing the concept of Object-Oriented Design, we tried to explain how individual design process could be programmed in terms of *object*s and step *object*s. Without the burden of understanding a programming language, designers create easily their own programming of what they design, and investigate their algorithm of form-making process. Also, providing a way of recording building information of each design artifact in a database, we suggested the model of an architectural reference for studying and developing the design artifact.

The further development should regard parametric transformation as one of basic transformations for affording more flexible design. In addition, for achieving more sophisticated design result, a way of combining design itself with information of other design disciplinary areas, which are history, environment, structure, urban planning and so on, in the data structure is needed.

6. ACKNOWLEDGEMENTS

In memory of his intellectual excellence and humour, I contribute this paper to my mentor and friend, the Late Emmanuel-George Vakalȯ.

[8] http://www-personal.umich.edu/~egvakalo/nsgc/teaching/design.htm

7. REFERENCES

Abelson and Sussman, 1996, *Structure and Interpretation of Programs*, The MIT press, Cambridge, Massachusetts, p. 79-93.

Agarwal, M., Cagan, J & Constantine, K.G.,1998, "Influencing Generative Design Through Continuous Evaluation: Associating Costs with the Coffeemaker Shape Grammar" *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)* Vol. 13 Number 4, p. 253-275.

Alexander, C., 1967, *Notes on the Synthesis of Form.* , Harvard University Press, Cambridge, Massachusetts.

Chen, K and Owen ,C. , 1997, "Form Language and Style Description", *Design Studies* 18, Elsevier Science Ltd, Great Britain, p.249-274.

Chomsky, N., 1978, *Syntactic structures*, The Hague : Mouton.

Dewey, J. , 1986, *Logic-The Theory of Inquiry: Volume12, The Later Works of John Dewey, 1925-1953.* , Southern Illinois University Press , Carbondale.

Koning, H. and Eizenberg. , 1981, "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses ", *Environment and Planning B: Planning and Design,* Vol.8, p. 295-323.

Lulushi, A., 1998, *Inside Oracle Designer/ 2000*, Prentice Hall, Inc., New Jersey.

Maruyama, H., Tamura, K and Uramoto, N. , 1999, *XML and JAVA*, Addison- Wesley Longman Inc. , Massachusetts, p. 14-30.

Mitchell, W.J. , 1989, *The Logic of Architecture,* MIT press, Cambridge, Massachusetts.

Papanek, V., 1984, *Design for the Real World.* Van Nostrand Reinhold.

Perry, M , 1998, "Coordinating Joint Design Work: the Role of Communication and Artefacts", *Design Studies* 19, Elsevier Science Ltd, Great Britain, p. 273-288.

Piazzalunga, U. and Fitzhorn, P. I., 1998, "Note on a Three-dimensional Shape Grammar Interpreter", *Environment and Planning B: Planning and Design* Vol25, p. 11-33.

Peirce, C.S. , 1966, "How to Make Our Ideas Clear", in *Selected Writings of Charles S. Peirce(Values in a Universe of Chance)*, edited by Philip P.W, Dover Publications, New York.

Ramakrishnan, R. , 1998, *Database Management Systems* , WCB/McGraw-Hill, INC. Boston, Massachusetts, p. 614-645.

Reese, G. , 1997, *Database Programming With JDBC and JAVA* , O'Reilly & Associates, INC. Cambridge, Massachusetts, p. 139-169.

Soutou, C. , 2000, "Modeling relationships in object-relational database", *Data & Knowledge Engineering*, vol. 36, p. 79-107.

Stiny, G. and Gips, J., 1972, "Shape Grammars and the Generative Specification of Painting and Sculpture" in C V Freiman (Ed) *Information Processing 71,* Amsterdam, North-Holland, p. 1460-1465.

Stiny, G. and Mitchell, W.J., 1978, "The Palladian Grammars " , *Environment and Planning B5: Planning and Design* Vol1, p. 5-18.

Stiny, G., 1980, "Kindergarten Grammars: Designing With Froebel's Building Gifts" , *Environment and Planning B: Planning and Design* Vol7, p. 409-462.

Stiny, G. and Mitchell, W.J., 1981, "The Grammar of Paradise ", *Environment and Planning B7: Planning and Design* Vol2, p. 209-226.

Stiny, G., 1981, "A Note on the Description of Designs ", *Environment and Planning B: Planning and Design* Vol8, p. 257-267.

Stiny, G. and March, L. , 1981, "Design Machines", *Environment and Planning B: Planning and Design* Vol8, p. 245-255.

Stiny, G., 1992, "Weights", *Environment and Planning B: Planning and Design,* Vol9, p. 413-430.

Stiny, G., 1994, " Shape Rules: closure, continuity, and emergence", *Environmental Planning B: Planning and Design,* Vol21, p. 49-78.

Szkman, S , Racz J, Bochenek C, and Sriram, R. D , 2000, "A Web-based System for Design Artifact modeling", *Design Studies* 21, Elsevier Science Ltd, Great Britain, p. 145-165.

Waite, M , 1998, *Object-Oriented Design in Java*, Stephen Gilbert and BillMaCarty Corre Madera, California.

Yair, K., 1999, "Design Through Making: crafts knowledge as facilitator to collaborative new product development", *Design Studies* 20, Elsevier Science Ltd, Great Britain, p. 495-515.

Zeisel J., 1981, *Inquiry by Design*, Brooks/Cole Publishing Company, Monterely, California.