Proceedings of ICAD2004
The Third International Conference on Axiomatic Design
Seoul – June 21-24, 2004
ICAD-2004-14

SOFTWARE DEVELOPMENT OF A SEQUENTIAL ALGORITHM WITH ORTHOGONAL ARRAYS (SOA) USING AXIOMATIC DESIGN

Jeong-Wook Yi
yijwook@ihanyang.ac.kr
Post Doctoral Fellow,
Center of Innovative Design Optimization Technology,
Hanyang University
17, Haengdang-Dong, Sungdong-Gu, Seoul, 133-791,
Korea

Gyung-Jin Park
gjpark@hanyang.ac.kr
Department of Mechanical Engineering,
Hanyang University
1271, Sa-1-Dong, Ansan, Gyeonggi-Do, 425-791,
Korea

ABSTRACT

The automatic design of engineering systems has been accomplished by the development of engineering optimization techniques. The methods find design solutions that minimize the cost function while given constraints are satisfied. The types of design variables are classified into continuous and discrete ones. Generally, available designs are discrete in design practice. However, optimization has been developed to determine designs in a continuous space. In recent research, a sequential algorithm using orthogonal arrays (SOA) has been proposed for design in a discrete space.

A software system is developed for the developed algorithm according to the process in the V-model, which is proposed as a design process for the object-oriented software system. In the conceptual design of the software, the axiomatic approach is utilized. Functional requirements (FRs) and design parameters (DPs) of the software are defined according to the Independence Axiom and decomposed by the zigzagging process. The objects in object-oriented programming (OOP) can be generated by analysis of the full design matrix and each object is coded as a class. The design results are discussed.

Key Words: Axiomatic Design, Independence Axiom, OOP (Object-Oriented Programming), SOA (Sequential algorithm with Orthogonal Arrays), Orthogonal Arrays

1 INTRODUCTION

Engineering optimization is an automatic design technique minimizing the objective function over satisfying constraint conditions after formulating design variables and requirements of main performance [Arora, 1989; Vanderplaats, 1984]. Recently, due to advances of the finite element method (FEM) and computer techniques, the method is being applied actively to the structural design field. The design variables are classified into continuous and discrete ones. Available designs are generally discrete in design practice. In structural design, design variables should be determined frequently among some given values or standard parts [Haftka, 1990; Arora, 1994]. But optimization has been developed to determine designs in continuous space. Therefore, it is needed to develop the method which can be applied to problems with discrete design variables.

Various algorithms for optimization in discrete design space have been proposed. Algorithms such as the branch and bound (B&B), the simulated annealing, the genetic algorithm and the taboo search serve as examples [Gutkowski, 1997; Osman, 1996; Tseng, 1995; Kirkpatrick, 1983; Gen, 2000]. These methods are applied to various areas, however, they are quite costly for large-scale problems due to many function evaluations. Recently, an efficient algorithm is proposed by Lee, et al [2003]. This algorithm, called SOA (sequential algorithm using orthogonal arrays), can decrease the function evaluations efficiently by using orthogonal arrays.

Axiomatic design (AD) provides a framework for general designs. Design involves a continuous interplay between what we want to achieve and how we want to achieve it. Axiomatic design helps a designer create a method for solving the synthetic problem. It consists of two axioms which are the Independence Axiom and the Information Axiom. When a design process is defined according to the Independence Axiom, the design can be performed sequentially without feedback [Suh, 1990, 2001; Park, 2005]. Since the design process is well organized, a software system can be easily developed for automatic design by using the axiomatic approach. Various software systems have been developed by axiomatic design [Do, 2001; Park, 1999]. Recently, the axiomatic approach is adopted for object-oriented programming (OOP) [Do, 1999]. Software design is carried out by the AD framework. It is noted that the software design and the design process are almost identical and the design process is automated by the software system.

The design process for embodying a sequential algorithm using orthogonal arrays (SOA) is analyzed and constructed based on the axiomatic design. The software system is developed by the object-oriented programming language. The V-model has been proposed for the idea of object-oriented software design with an AD framework [Do, 1999]. The system is designed by using the V-model and coded according to the design.

2 SEQUENTIAL ALGORITHM USING ORTHOGONAL ARRAYS (SOA)

A matrix experiment for DOE consists of a set of experiments. The settings of design variables are defined to obtain the characteristics. After all of the experiments are conducted, ANOM (analysis of mean) is performed to determine the optimum levels [Park, 2002]. Since the DOE with orthogonal arrays uses discrete values of design variables, discrete design can be easily carried out.

When orthogonal arrays are properly selected, the minimum number of experiments would have an effect that substitutes the full factorial experiments. If interaction among the design variables is strong, the interaction should be considered in choosing the smallest size of an orthogonal array. However, it is not easy to comprehend the interactions in structural design. In this research, the effect of the interaction is disregarded.

For discrete designs without interaction or with mild interaction, the Taylor series method can be utilized. In the Taylor series method, perturbations of the design variables are made forward and backward one at a time from the current design. It can be superior to the suggested method for that kind of problem. However, in problems with strong interaction, the Taylor series method depends on only the current design point, which may lead to the solution far from the real optimum. On the contrary, the method using the orthogonal array offers a closer solution to the real optimum since it determines the optimum level using the average effect of each design variable.

An algorithm has been proposed using the estimation values by ANOM [Park, 2002]. Since the algorithm uses orthogonal arrays, the number of function evaluations can be decreased greatly. Also, it can obtain a local discrete optimum by sequentially moving the searching region. The overall flow of the algorithm is illustrated in Fig. 1. The steps of the proposed algorithm are as follows:

Step 1: Definition of the problem

An optimization problem is defined through the identification of the design variables, an objective function, and constraints. The factors in DOE are equivalent to the design variables in optimization. The levels and the characteristic function in the DOE are regarded as the discrete values and the objective function in optimization, respectively. The candidate values of each design variable should be given for the discrete design. These values are considered as the levels in the matrix experiment.

Step 2: Selection of an orthogonal array

In this research, three-level experiments are adopted for finding a new design in an iteration. Thus, the minimum orthogonal array is selected according to the number of design variables. Minimum orthogonal array is utilized to save computational time. The minimum orthogonal array is the smallest one where all the design variables can be assigned to its columns.

Step 3: Arrangement of the levels for design variables

In this step, the discrete values of design variables are assigned to the columns of the orthogonal array selected in Step 2. The assignment can be arbitrarily performed, because the interactions are not considered.

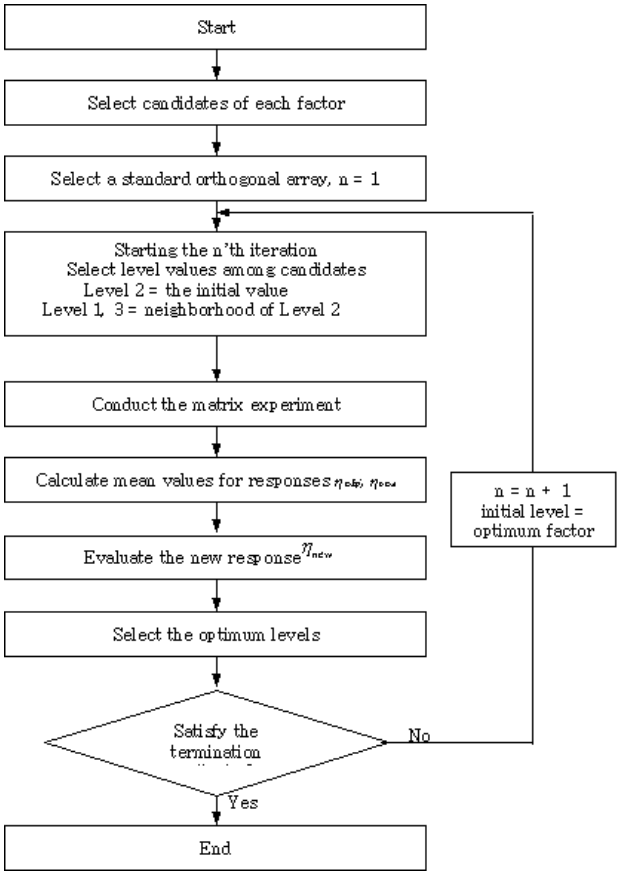

Fig. 1 Flow of sequential algorithm using orthogonal arrays (SOA)

The candidate values of design variables should be given before the design process is applied. In the initial design, an arbitrary discrete value can be selected. The selected value is assigned to the second level. From the neighboring values, the discrete value with one step larger than the second level is assigned to the first level while the one step smaller value is assigned to the third level. Safe design is important in structural design. Therefore, larger values are assigned to the first levels, because the first levels appear more in orthogonal arrays. If an initial design has the smallest or the largest from the candidates, the two successively increased or decreased discrete values are selected as the levels.

Step 4: Matrix experiment

The characteristic function is calculated for each row of the orthogonal array. Using the optimization formulation, the matrix experiment for each iteration can be expressed as follows:

$$\begin{aligned} &\text{Find} && \mathbf{b} && && (1a)\\ &\text{to minimize} && f(\mathbf{b}) && && (1b)\\ &\text{subject to} && g_i(\mathbf{b}) \le 0 && i=1,\ldots,l && (1c) \end{aligned}$$

where **b** is the design variable vector, $f(\mathbf{b})$ is the objective function and $g_i(\mathbf{b})$ is the constraint function, and l is the number of constraints.

Step 5 : Determination of a new design

The conventional DOE methods do not consider the constraints defined in design optimization. Thus, a new response called the characteristic function, $\hat{R}_{new}$ is defined to include the estimation value for the constraint violations as follows:

$$\hat{R}_{new} = \hat{f}(\mathbf{b}) + \hat{P}(\mathbf{b}) \quad (2a)$$

$$\hat{P}(\mathbf{b}) = s \times \sum_{i=1}^{m} \max[0, \hat{v}_i] \quad (2b)$$

where $\hat{P}(\mathbf{b})$ is the penalty function, $\hat{v}_i$ is the maximum violation of the estimation value of the i-th constraint, and s is the scale factor. The scale factor is imposed to emphasize the constraint violation. In this research, the scale factor is set to a value so that the order of the penalty function is one-order larger than that of the original objective function, $f(\mathbf{b})$.

Next, after sorting the estimation values $\hat{R}_{new}$ in increasing order, we can select the value and condition in the top as an optimum one. It produces new levels of design variables. The confirmation analysis with new levels should be carried out since it does not always guarantee the design feasibility or the statistical validity of additivity. The constraint feasibility is not always guaranteed due to the definition of the characteristic function. Furthermore, if interaction between design variables exists, the interaction can cause an unreasonable determination of the new levels. Thus, we compare the new levels evaluated by estimation values of full factorial with the best combination from the matrix experiments and the best levels are selected as the new levels in the iteration. In structural design, one experiment requires finite element analysis.

Step 6: Convergence criteria

The convergence criteria in the algorithm are as follows: The algorithm is terminated (1) if the optimum levels of each factor are all 2-levels in the current iteration, (2) if the number of iterations in which the responses at the new levels are consecutively not feasible is more than five. In case of termination criterion (1), the current design is a local optimum solution. If none of convergence criteria is satisfied, the design process returns to Step 3. In this step, the second levels for the design variables are replaced into former optimum levels.

3 DEVELOPMENT OF A SOFTWARE FOR EMBODYING SOA USING THE AXIOMATIC APPROACH

3.1 AXIOMATIC DESIGN FOR OBJECT-ORIENTED PROGRAMMING

Axiomatic design (AD) is the framework for a good design. It helps to create synthesized solutions that satisfy perceived needs through mapping between functional requirements (FRs) and design parameters (DPs). An FR is the goal to achieve and is defined in the functional domain, while a DP is determined in the physical domain as the means to achieve the goal. Mapping is a process to choose a relevant DP in the physical domain, which satisfies the corresponding FR in the functional domain. Designers begin the design from comprehensive FRs. A design can decompose FRs into many hierarchies. But the decomposition of FRs must be carried out at the same time with the decomposition of DPs. The zigzagging between FRs and DPs is necessary because the two sets of each level are connected and mutually dependent.

Axiomatic design consists of two axioms which are the Independence Axiom and the Information Axiom. The first axiom tells us about the selection of FRs and DPs. The second axiom shows a quantitative method of judging which design is more desirable. The two axioms present the most fundamental means needed to choose the best design.

For a design to be acceptable, the design must satisfy the first axiom. A design matrix (DM) is defined to pursue the relationship between FRs and DPs as following:

$$\{FRs\} = [A]\{DPs\} \quad (3)$$

where $\{FRs\}$ is a vector for FRs, $\{DPs\}$ is a vector for DPs and $[A]$ is a design matrix.

When the Independence Axiom is satisfied, the design matrix takes the form of a diagonal matrix or a triangular matrix. A diagonal matrix represents a perfectly uncoupled design and is the most desirable form. A triangular matrix represents a decoupled design. This form of design is also a proper design, but the DPs need to be determined in a specific order. The third form is the coupled design where some diagonal elements are not zero in the design matrix. This type of design is undesirable because when a DP is modified, multiple FRs are changed. Thus, the Independence Axiom is not satisfied.

The Information Axiom is related to the complexity of a design, and implies that the simpler design is the better one. Generally, information content is quantitatively defined by the probability of success. The Information Axiom is utilized to select the best one out of multiple designs which satisfy the Independence Axiom. The information content is not yet defined rigorously in software development. Therefore, only the Independence Axiom is employed in this research.

System modules should be independently constructed to design an efficient software system [Roger, 1997]. It is needed to introduce the axiomatic design approach so that modules in the functional domain can be maintained independently in the physical domain. Do and Park have developed a software for the glass bulb design with a conventional language using axiomatic design [Do and Park, 2001]. Recently, object-oriented programming (OOP) is utilized with the axiomatic approach and the V-model is proposed as a design process for the object-oriented software system [Do and Suh, 1999]. In the V-model, the first step is to build the hierarchy, which follows the top-down approach, and is the generation of the full design matrix table for module definition. The second step is to build the object-oriented model with a bottom-up approach. As illustrated in Fig. 2, several steps are needed.

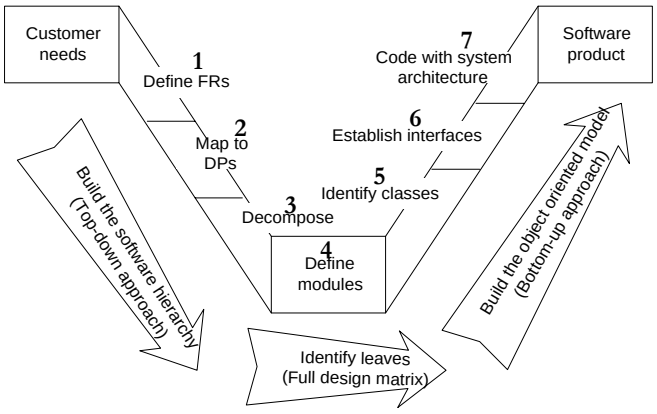

Fig. 2 Axiomatic design process for object-oriented software system (the V-model)

Table 1 FRs of the top level

	FRx	DM					DPx
P	Develop the SOA software						SOA software
1	Construct the user's environment	**X**	**O**	**O**	**O**	**O**	Data set of software environment
2	Construct the input condition of design	**X**	**X**	**O**	**O**	**O**	Input data set of the design
3	Conduct the sequential process for optimum solution	**O**	**X**	**X**	**O**	**O**	Condition data for iterative process
4	Show the results	**O**	**X**	**X**	**X**	**O**	Result data set and output resource
5	Manage the information for software	**X**	**X**	**X**	**O**	**X**	Information data

Table 2 FRs of the first level for FR1x

	FR1x	DM1		DP1x
P	Construct the user's environment			Data set of software environment
1	Construct the working folder	**X**	**O**	Data set of setting the option information
2	Construct the information of analyzer	**O**	**X**	Data set of analyzer

Table 3 FRs of the second level

	FR12x	DM12						DP12x
P	Construct the information of analyzer							Data set of analyzer
1	Define the name of analyzer	**X**	**O**	**O**	**O**	**O**	**O**	Name of analyzer
2	Search the position of analyzer	**O**	**X**	**O**	**O**	**O**	**O**	Position of analyzer
3	Define the file name of analyzer	**O**	**X**	**X**	**O**	**O**	**O**	File name of analyzer
4	Define the extension of analyzer	**O**	**X**	**O**	**X**	**O**	**O**	Extension of analyzer
5	Define the extension of output file	**O**	**O**	**O**	**O**	**X**	**O**	Extension of output file
6	Define the format type of input file	**O**	**O**	**O**	**O**	**O**	**X**	Format type of input

The system, which can embody a sequential algorithm using orthogonal arrays (SOA), is designed by using the V-model and coded according to the design. We will describe the development process of the software system according the design step in the V-model. Parts of the development will only be introduced because the entire process is too long to describe.

3.2 DEFINITION OF FRS FOR THE SYSTEM AND DECOMPOSITION (STEPS 1, 2 AND 3)

First, the process for SOA is analyzed from an axiomatic viewpoint. The SOA software has to conduct the link with other commercial analysis packages and be composed of functions which users can easily use. Also, each iterative process has to be executed automatically and the results should be shown to users. These requirements are defined as engineering terms. As a result, FRs, DPs and their relations for the top level are defined as shown in Table 1. An FR is expressed by a function that the software system should achieve. Rhetorically, the sentence starts with a verb. DPs are data to achieve the FRs and they are the input and output of programs in software design. Rhetorically, they start with nouns. "X" in the design matrix means an algorithm to materialize the logical relations.

The design process is a decoupled one because the design matrix is triangular. Thus, the software design should be carried out according to the sequence that the design matrix indicates. And then each FRx is decomposed to FRx's based on the selected DPx. DPs in a certain level play roles for standards to decomposed FRs of the next lower level. It is noted that the independence of FRs should be maintained by appropriate selection of DPs.

As shown in Table 1, FR1 is "construct the user's environment." And DP1 is input and output data of software environment such as working folder and information of a commercial analyzer. Environmental data is a set consisting of many data. Therefore, it is defined as a structure type in the C-language. FR2 is "construct the input condition of design." DP2 for the satisfying function of FR2 is input data set of the design, and FR2 also needs environmental data of DP1, which is expressed by "X" in Table 1. For example, for performance of FR2, the function of FR1 is needed. Then, since the name of the working folder involved in the environmental data set has to be used, information of DP1 must be used. Similarly, FR3, FR4 and FR5 are defined and DP3, DP4 and DP5 are defined accordingly.

From DP1, the detailed operations of FR1 can be defined. That is, various data constructions should be made for data for

running software and data of analyzer. Therefore, FR1 is decomposed into FR11 and FR12 as shown in Table 2. The type of design is an uncoupled design in the lower level. Therefore, each function of FRs can be performed independently. DP12, "Data set of analyzer," consists of name, position, file extension, type of data, etc. FR12 is decomposed into six FRs from FR121 to FR126 based on each DPs, as shown in Table 3.

The decomposition is continued up to the minimum unit of the algorithm. The flow of the software system is the same as that of the design process except for the options of the software system and data management.

3.3 Definition of Modules and Identification of Objects (Steps 4 and 5)

The entire full design matrix is established from the zigzagging process of decomposition. The full design matrix is exploited for definition of software modules and objects. Fig. 3 illustrates the full design matrix. The rows of the matrix represent FRs and the columns represent DPs. In software design FRs and DPs are modules and input/output for functions, respectively.

The rectangular matrices with thick lines represent independent sub matrices. Each FR is defined as a module and each module is defined in the functional domain while each DP is defined in the physical domain. Therefore, the design matrix shows the relationship between the functional domain and the physical domain.

In view of object-oriented programming (OOP), we link methods and attributes in an object into FRs and DPs and objects in Fig. 3 are defined. An object, which is represented by the rectangle with thick lines, consists of the methods in the row of the design matrix and the attributes of the column. Main module is defined by an object of "ExteriorPenalty" which is inherited by the "DOEMethod" object and involves objects such as "OptionDlg." In the lower level, the module of FR1 is performed by objects of "OptionDlg" and "AnalyzerInfoDlg." FR2 consists of objects of "InputInfoSettingDlg," "FactorInfoDlg" and "OATableSettingDlg." Similarly, FR3, FR4 and FR5 consist of other objects as shown in Fig. 3.

In the lower level, the module of FR1 can be decomposed into FR11 and FR12. FR11 sets a working folder and FR12 defines the information of an analyzer for user, as illustrated in Fig. 4. The function of FR12, which is decomposed into FR122, FR123 and FR124, can be performed by objects of "AnalyzerInfoDlg." An object of "DirSearchDlg" is involved in the object of "OptionsDlg." All the data related to these objects are involved in the object of the "ExteriorPenalty" which has the structure data of "Analyzer_Info," "Option_Info," etc. All the objects are defined in this way.

3.4 Establishment of Interfaces and Coding (Steps 6 and 7)

Classes are defined by the set of objects as illustrated in Fig. 5. The only one class for FR1 is presented in Fig. 5. Almost all of the names of the classes are the same. Class "OptionsDlg" is defined from the relation of "Aggregation" for the two classes,

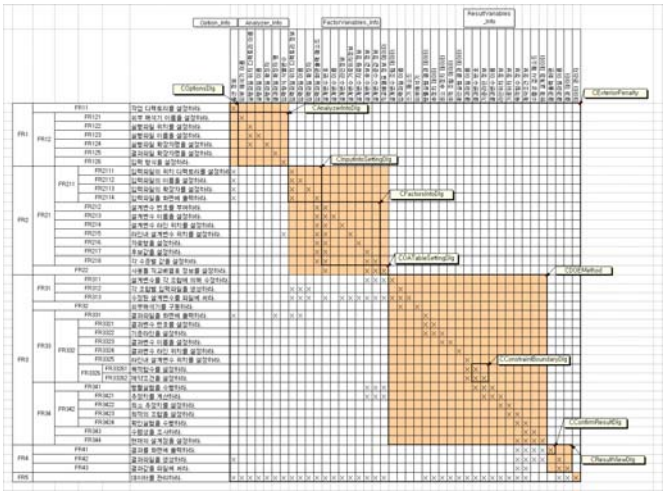

Fig. 3 Full design matrix

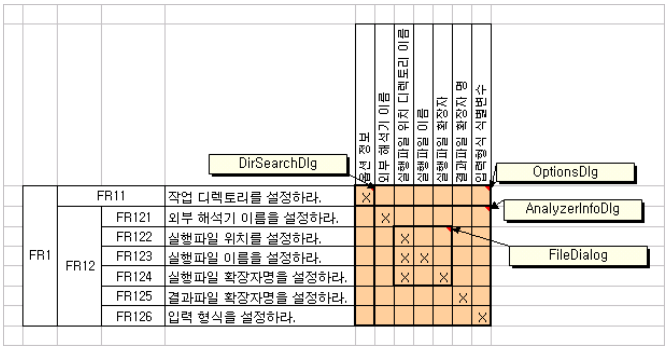

Fig. 4 Design matrix for FR1

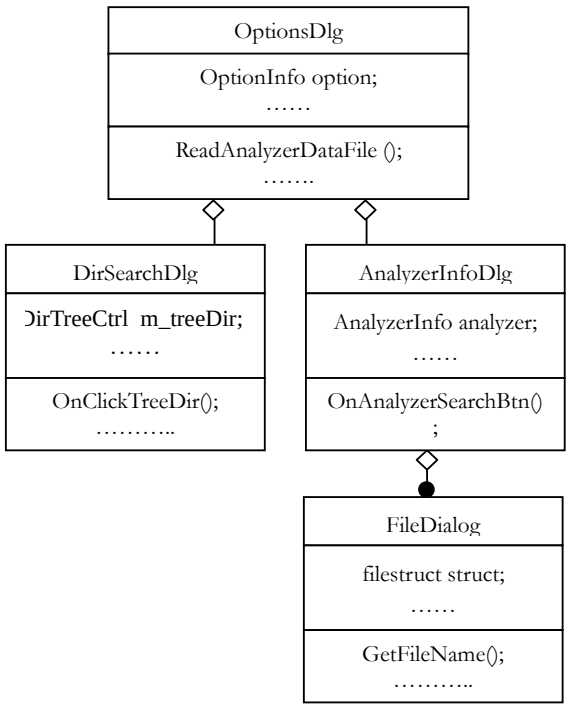

Fig. 5 Class diagram for FR1

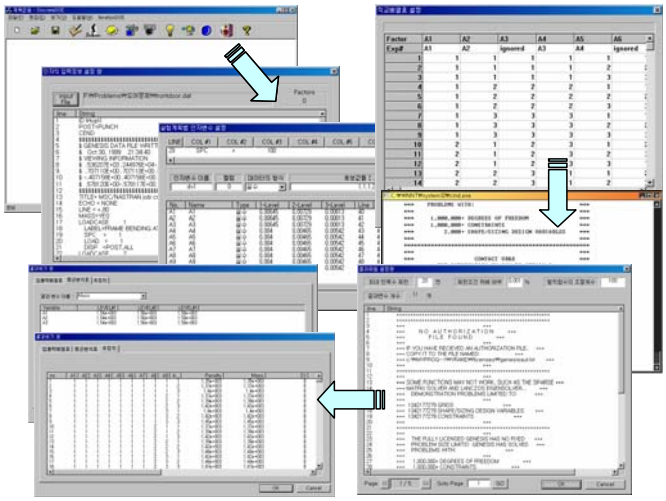

Fig. 6 Screen of the developed software (SOA)

"DirSearchDlg" and "AnalyzerInfoDlg." Class "OptionsDlg" has a function of input for constructing the configuration file. It is automatically executed and fixed when the system starts. Class "AnalyzerInfoDlg" handles the data for management of other commercial analysis package or user defined execute file. Any classes defined in this step are used as the new objects in another process. For example, in Fig. 5, class "FileDialog" is used for the function of FR211, which sets the path of an input file for an analyzer.

Using the above process, a software system is coded. The overall menus are illustrated in Fig. 6. After setting design variables and selecting the table of orthogonal arrays, the system iteratively conducts experiments according to the selected orthogonal arrays and shows the optimal condition.

4 CONCLUSIONS

A design software system is developed to perform the sequential algorithm using orthogonal arrays (SOA). The system is systematically designed by using the axiomatic approach. The V-model for object-oriented programming is adopted in this process and coding is conducted based on the software design. It is found that the axiomatic approach makes software development quite handy. The process is incorporated exactly in the flow of the software system with various menus. For strength analysis, the system is interfaced with a finite element analysis system.

In the future, application to commercial products will be made because the time for software development is considerably reduced due to the axiomatic approach.

5 ACKNOWLEDGMENTS

This research was supported by the Center of Innovative Design Optimization Technology, which was funded from the Korea Science and Engineering Foundation. The authors are thankful to Mrs. MiSun Park for her correction of the manuscript.

6 REFERENCES

[1] Arora, J.S., *Introduction to Optimum Design*, New York: McGraw-Hill Book Company, 1989. ISBN 0-07-100123-9

[2] Arora, J.S., Huang, M.W., "Methods for Optimization of Nonlinear Problems with Discrete Variables: A Review," *Structural Optimization*, Vol. 8, pp. 69-85, 1994.

[3] Do, S.H., Park, G.J., "Application of Design Axioms for Glass Bulb Design and Software Development for Design Automation," *Journal of Mechanical Design of the ASME*, Vol. 123, No. 3, pp. 322-329, 2001.

[4] Do, S.H., Suh, N.P., "Systematic OO Programming with Axiomatic Design," *Computer*, Vol. 32, No. 10, pp. 121-124, 1999.

[5] Gen, M., Cheng, R., *Genetic Algorithms and Engineering Optimization*, New York: John Wiley & Sons, Inc., 2000.

[6] Gutkowski, W., *Discrete Structural Optimization*, New York: Springer-Verlag, 1997.

[7] Haftka, R.T., Gurdal Z., Kamat, M., *Elements of Structural Optimization*, Dordrecht: Kluwer Academic Publishers, 1990.

[8] Kirkpatrick, S., Gelatt, C.D., Jr., M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671-680, 1983.

[9] Lee, K.H., Yi, J.W., Park, J.S., Park, G.J., "An Optimization Algorithm Using Orthogonal Arrays in Discrete Design Space for Structures," *Finite Elements in Analysis and Design*, Vol. 40, pp. 121-135, 2003.

[10] Osman, I.H., Kelly, J.P., *Meta-Heuristics: Theory & Applications*, Boston: Kluwer Academic Publishers, 1996.

[11] Park, G.J., *Analytic Methods in Design Practice*, Springer-Verlag, in preparation, 2005.

[12] Park, G.J., Do, S.H., Suh, N.P., "Design and Extension of Software Systems Using the Axiomatic Design Framework," *Transactions of the Korean Society of Mechanical Engineers (A)*, Vol. 23, No. 9, pp. 1536-1549, 1999 (in Korean).

[13] Park, S.H., *Modern Design of Experiments*, Minyoung-Sa, 2002 (in Korean).

[14] Roger, SP., *Software Engineering: A Practitioner's Approach*, 4th Ed. NY: McGraw-Hill, 1997. ISBN 0-07-052182-4

[15] Suh, N.P., *The Principles of Design*, NY: Oxford University Press, 1990. ISBN 0-19-504345-6

[16] Suh, N.P., *Axiomatic Design: Advances and Applications*, NY: Oxford University Press, 2001. ISBN 0-19-513466-4

[17] Suh, N.P., Sekimoto S., "Design of Thinking Design Machine," *Annals of the CIRP*, Vol. 39, No. 1, pp. 145-148, 1990.

[18] Tseng, C.H., Wang, L.W., Ling, S.F., "Enhancing Branch-and-Bound Method for Structural Optimization," Journal of Structural Engineering, *ASCE*, Vol. 121, pp. 831-837, 1995.

[19] Vanderplaats, G.N., *Numerical Optimization Techniques for Engineering Design*, NY: McGraw-Hill Book Company, 1984.

