USING DSM AND fmGA TO DETERMINE THE OPTIMAL DESIGN PROCESS FOR ENGINEERING DESIGN PROJECTS

Chung-Wei Feng, Associate Professor
Dept. of Civil Engineering
National Cheng Kung University
No.1, Ta-Hsueh Road, Tainan 701, Taiwan
cfeng@mail.ncku.edu.tw

Yu-Chuan Yeh, Ph.D Candidate
Dept. of Civil Engineering
National Cheng Kung University
No.1, Ta-Hsueh Road, Tainan 701, Taiwan
whaly.yeh@msa.hinet.net

Abstract: The quality of the design has been recognized as one of the key factors to the success of the engineering project. However, engineers need to repeatedly perform design tasks, such as processing information, to produce an appropriate design result. Such a design process involves many activities and lead into one or several iterative loops. Consequently, the sequence of performing design activities should be properly planned to avoid unnecessary loops. Design Structure Matrix (DSM) is one of the widely used methods to describe the design process. By employing DSM, the relationships and the sequence of the design activities can be identified. In this research, several principles of evaluating the design process are developed. In addition, fast messy Genetic Algorithm (fmGA) is employed along with the proposed evaluation principles and DSM to determine the optimal design process. Results show, based on the proposed evaluation principles, the impact of the iteration loops can be clearly identified. Furthermore, with the proposed optimization model and its computer implementation, engineers can find the optimal design process for the large-scale design project efficiently and effectively.

Keywords: Design Planning, Optimization, DSM, fmGA

1. INTRODUCTION

The design phase is one of the most important stages of the engineering project. To effectively and efficiently perform the activities within the design phase, the project manager must carefully plan the design process. Therefore, project managers usually have to first understand the relationships between activities within the design process and then develop a suitable design process based on these relationships. However, the information used in the design activities are usually interdependent, engineers have to make some assumptions before performing related design tasks. For example, to design a certain structure of the building, some presumed loading conditions must be developed first. These conditions will further be checked and modified if needed during the process of designing the structure. Because of the interrelationships between activities, the design teams may be trapped within the loops of design activities due to poor information flow and the inappropriate allocation of resources. Consequently, the size of repeating loops, which is number of activities within the loop, plays an important role in terms of the efficiency and effectiveness of the design process. There have been many techniques developed to determine the optimal design process. Among them, Design Structure Matrix (DSM) is one of the widely used techniques to identify the relationships between activities within the design process. By employing DSM, the relationships between activities and the effectiveness of the design process can be identified. However, since the number of activities within the design process could be numerous, a searching algorithm should be employed to determine the optimal design process. In this research, several principles to evaluate the design process are developed. In addition, fast messy Genetic Algorithm (fmGA) is employed along with the proposed evaluation principles and DSM to determine the optimal design process. Results show, based on the proposed evaluation principles, the impact of the iteration loops can be clearly identified. Furthermore, with the proposed optimization model and its computer implementation, engineers can find the optimal design process for the large-scale design project efficiently and effectively.

2. DESIGN STRUCTURE MATRIX

Design structure matrix (DSM) was proposed by Steward [1] in 1981. Many researchers have successfully applied DSM to solve the complex design process in the last decade [2][3][4][5][6][7][8]. DSM first divides the design project into *n* individual activities in an *n* x *n* matrix which contains *n* rows and columns. Information links among individual activities are clearly shown by the systematic mapping, regardless of number of activities. Figure 1 is an example of DSM.

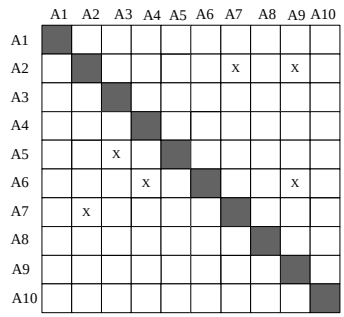


Figure 1. Design Structure Matrix

DSM is also referred as the Dependency Structure Matrix reflecting its application outside design. Rogers et al. [9] proposed a distinct format of DSM, as shown in Figure 2, to identify the information flow and the interrelationships among activities. In this format of DSM, interesting and useful information can be discovered. Couplings in the upper triangle portion of the DSM represent feedforward data; couplings in the lower triangle part of the matrix represent feedback data. A feedback implied an iterative process in which an initial guess must be made. In addition, crossover occurs when feedback from one activity to another activity without exchanging data through the intersection. Iteration loops exist when activities are within the feedfowards and then feedbacks situation, for example, activity G and H form an iteration loop

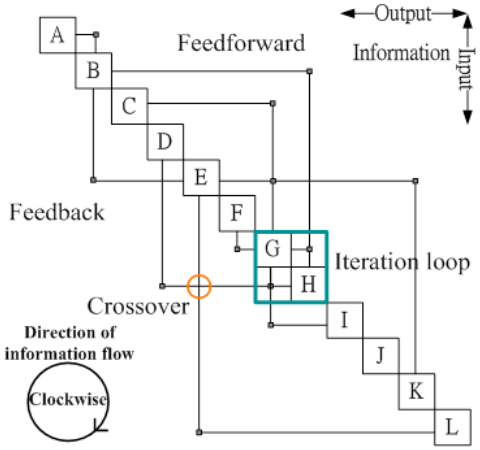


Figure 2. Another format of DSM

Feedforwards, feedbacks, crossovers, and iteration loops are good indications for evaluating design processes. This research employs the

3. FAST MESSY GENETIC ALGORITHMS

The fast messy genetic algorithms (fmGA) were developed by Goldberg et al. in 1993 [10]. Unlike the well-known simple genetic algorithms (sGA)", which uses fixed-length strings to represent possible solutions, the fmGA applies messy chromosomes to form strings of various lengths. The fmGA can efficiently find the optimal solution of the large-scale permutation problem [11]. In addition, the GA-based approach has been known for its flexibility in hybridizing with other methodologies to obtain better solutions [11]. Since the departing time of each truck assigned to various construction sites is obtained by simulating the delivering process, the fmGA is adopted to integrate with the simulation methodology to find the dispatching sequence of the optimal dispatching schedule in this study. The elements and the process of the fmGA are briefly described in the following:

3.1 messy representation

In the fmGA, genes of a chromosome are represented by the pair (allele locus, allele value), in which allele locus indicates the position of the gene and allele value represents the value of the gene in that position. For example, two messy chromosomes ((5 0)(1 0)(3 1)(4 1)(2 1)(4 0)(5 1) and (3 1)(1 1)(5 0) may be both equivalent to the binary string 01110. As the above example shows, messy chromosomes may be "over-specified" and "underspecified" in terms of encoding bid-wise strings. Chromosome S1 is an over-specified string which has two different values in the positions of genes 1, 4, and 5. To evaluate the over-specified chromosome like S1, the string may be scanned from left to right with the first-come-first-serve rule. On the other hand, for evaluating the underspecified chromosome, such as S2, the competitive template is used. The competitive template is a problem-specified and fixed-bit string that is randomly generated or the solution found so far within the searching process.

3.2 messy operators

Messy operators that include the cut-splice operator and the mutation operator are used as genetic operators in the fmGA. The cut-splice operator, similar to the crossover operator in the sGA, is used to recombine different strings to create new strings. For example, as shown in Figure 3, a cut at point 3 would result in strings ((2 0)(5 0)(3 1)) and ((6 0)(5 1)). The splice operator joins two strings with a specified splice probability Ps. For example, as shown in Figure 3, two strings ((2 0)(5 0)(3 1)(6 0)(5 1)) and ((1 1)(2 1)(4 1)) would be recombined to ((2 0)(5 0)(3 1)(4 1)) and ((1 1)(2 1)(6 0)(5 1)) after being cut and spliced.

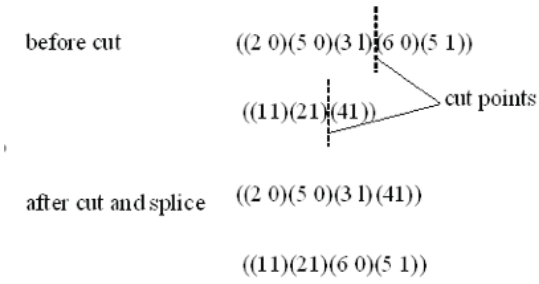


Figure 3. Cut-splice operator

The mutation operator perturbs the allele values of the messy chromosome by switching them from 1 to 0 and vice versa with a predefined probability Pm, which is similar to the mutation operator used in the sGA.

3.3 organization of the fmGA

There are two loops, the outer loop and the inner loop, within the fmGA. The outer loop iterates over the order *k* of the processed Building Blocks (BBs). Every cycle of the outer loop is denoted as an era. When a new era starts, the

inner loop which includes the initialization phase, the primordial phase, and the juxtapositional phase is invoked. The goal of the initialization phase is to create a population of strings containing all possible BBs of the order *k*. The primordial phase filters out the "bad" genes not belonging to BBs so that the population encloses a high proportion of "good" genes belong to BBs. Two operations, the building-block filtering and the thresholding selection, are performed within the primordial phase. In the juxtapositional phase, those good genes are combined by using the selection and the messy operators to form a high quality generation which perhaps contains the optimal solution. When the inner loop of the fmGA terminates, the outer loop of the fmGA begins with processing the BBs of order k+1. In addition, the competitive template is replaced by the best solution found so far, which becomes the new competitive template for the next era. The whole process is repeated until the maximum number required k_{max} is reached. The organization of the fmGA is shown in Figure 4.

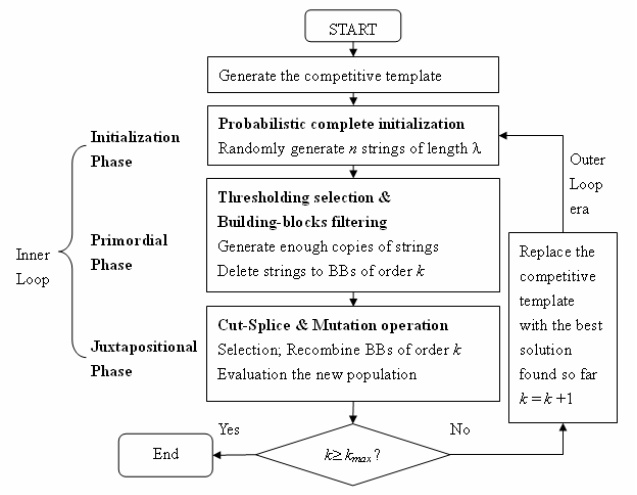


Figure 4. Organization of the fmGA

4. THE POPOSED MODEL

In this section, the proposed model that includes the evaluation of the design process and the application of the fmGA is presented.

4.1 Evaluation principles

In this research, the design process is evaluated according to three steps. The evaluation process is described as follows:

Step one: Defining information factors (IF). In this research, the IF is used to determine the level of the information dependency between two information related activities. The IF can be treated as the weight between 0 and 1. In addition, from the result of Austin [4], IF is determined according to (1) how dependent the activity is on the information; (2) how sensitive the activity is to the change of the information; and (3) how easily the information can be estimated. The larger the IF is, the more important the information flow is.

Step 2: Identifying feedbacks. According to the definition of feedback, the feedbacks of the DSM are identified and will be used to calculate the feedback factor for further analysis. Figure 5 is a DSM with six feedbacks.

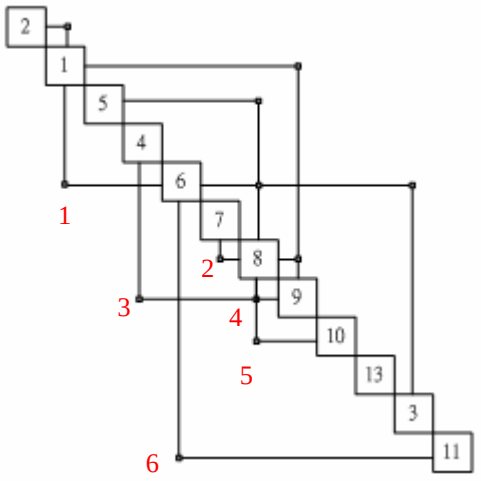


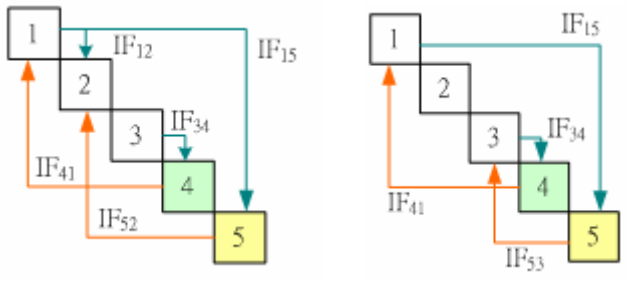
Figure 5. A DSM with 6 feedbacks

Step 3: Calculating feedback factor (FF). After the feedbacks are identified, the FFs can be calculated. First, the information flow trees have to be identified. The rules for developing information flow tree are stated as follow:

1. Initial point of the information flow tree is the earliest activity that feedbacks information.
2. Information flow three grows according to information links within the DSM. In addition, only the activities that are performed before the initial point are the candidate nodes could grow.
3. Information flow tree stops developing when (1) the current node has no further information dependent activities (2) the growing path is repeated.

Figure 6 shows two examples to demonstrate the development of the information flow three. Figure 7 presents the information flow trees of the Figure 6. In Figure 6 (a), only activities 4 and 5 have feedbacks to other activities. Therefore, there are two the information flow trees start from activity 4 and 5, respectively. In Figure 7(a), the information flow tree starts from activity 4 develops to activity 1 and then activity 2. This information flow tree then stops growing because activity 2 has no succeeding information related activities. Similarly the information flow tree starts from activity 5 develops to activity 2 and then stops In Figure 6 (b), the information flow tree stars from activity 4 and 5, respectively. The information flow three starts from activity 4 and grows to activity 1 and then stops since activity 5 is not the candidate nodes, as shown in

Figure 7 (b). On the other hand, the information flow tree starts with activity 5 and grows to activities 3, 4, 1, and then stops at activity 5 because of the growing path is repeated.



(a) (b)

Figure 6. DSM

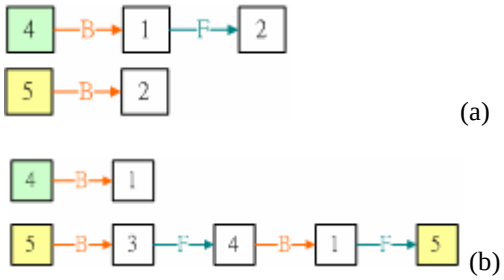


Figure 7. Information flow threes

As the information flow tree can be identified, the FF can be calculated by incorporating the IFs on the information flow tree. The feedback factor of the information flow tree is the sum of the IFs on the information flow tree and IFs on the feedbacks if the information flow tree has iterative loops.

$$FF_{ij} = \sum_{IF \in \text{information flow tree}} IF_{mn} + \sum_{\substack{IF \in \text{feedbacks,} \\ \text{if iterative loop} \\ \text{exists}}} IF_{mn} \quad \text{.....Eq.1}$$

where

i is the initial point and *j* is the end point of the information flow three.
m and n are activities linked by information dependency.

In addition, the total feedback factor of the design process is defined as the sum of all feedbacks, as shown in Eq. 2.

$$TFF = \sum_{FF \in feedbacks} FF_{ij} \quad \text{……..……..Eq. 2}$$

For example, the FF of the Figure 7 (a) can be calculated by

$$FF_{41} = IF_{41} + IF_{12}$$
$$FF_{52} = IF_{52}$$

and

$$TFF = FF_{41} + FF_{52} = IF_{41} + IF_{12} + IF_{52}$$

If $IF_{41} = 0.5$; $IF_{12} = 0.7$; $IF_{52} = 0.3$

then

$$TFF = IF_{41} + IF_{12} + IF_{52} = 0.5 + 0.7 + 0.3 = 1.5$$

For the design process in Figure 6 (b),

$$FF_{41} = IF_{41}$$
$$FF_{53} = IF_{53} + IF_{34} + IF_{41} + IF_{15} + (IF_{53} + IF_{41})$$

and

$$TFF = FF_{41} + FF_{53} = 3IF_{41} + 2IF_{53} + IF_{34} + IF_{15}$$

If $IF_{41} = 0.3$; $IF_{53} = 0.5$; $IF_{34} = 0.2$; $IF_{15} = 0.6$

then

$$TFF = 3IF_{41} + 2IF_{53} + IF_{34} + IF_{15}$$
$$= 3 \times 0.3 + 2 \times 0.5 + 0.2 + 0.6 = 2.7$$

4.2 The application of the fmGA

The design process can be defined as the permutation of the activities in terms of the execution order; therefore the chromosome structure should be able to represent the all possible permutations of activities. Figure 8 shows the chromosome structure used in this research. Two numbers are applied to depict the messy gene. The first number, from the left side, determines the position of the gene in the coded string, and the second number represents the activity index. Since the messy string could be over-specified or underspecified, the possible permutations could all be represented by this chromosome representation.

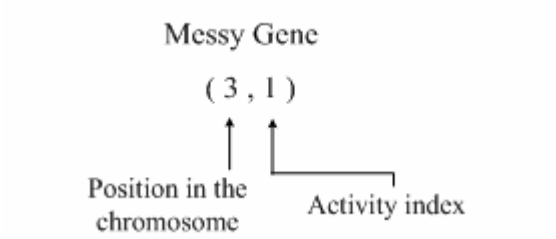


Figure 8. The chromosome structure

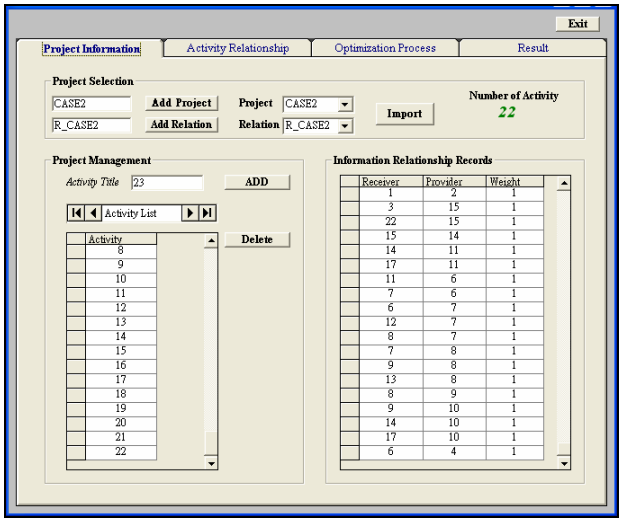
In addition, total feedback factors (TFF) of each design process is used for determine the fitness of the string generated by fmGA. Since the number of iterative loops should be minimized, the fitness function is defined as Eq. 3.

$$\text{Min } TFF = \sum_{FF \in feedbacks} FF_{ij} \quad \text{……Eq. 3}$$

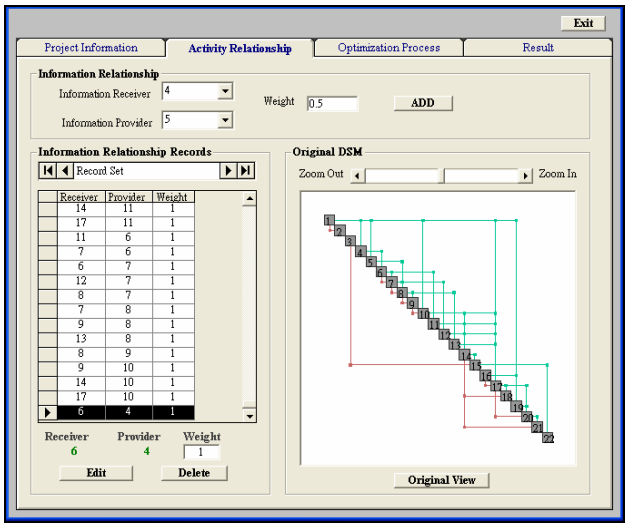
5. CASE STUDY

Along with development of the proposed model, a computer implementation called Design Project Process Optimizer (DPPO) is built. Figure 9 is the interface of the

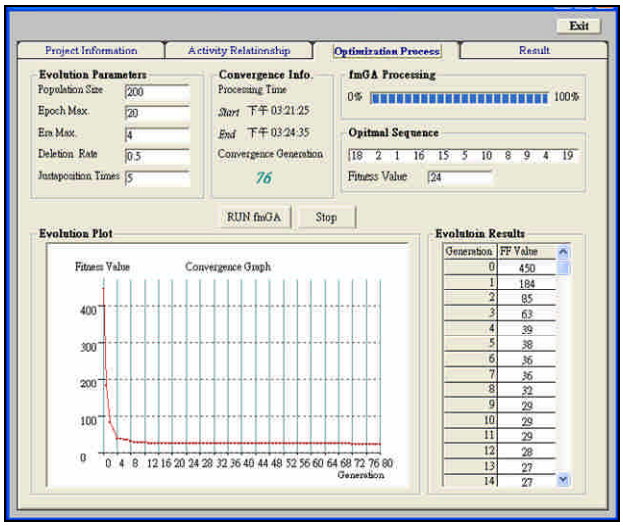
DPPO. As shown in Figure 9, there four modules which are Project Information, Activity Relationship, Optimization Process, and Result, within the DPPO. Project Information, as shown in Figure 9 (a), allows users to enter activity-related information of different projects. Activity Relationship, as shown in Figure 9 (b), provides users to input the information links between activities. Optimization Process, as shown in Figure 9 (c), requires users to input the parameters of fmGA. Users can then perform the optimization of the design process when the parameters are input. In addition, the convergence process as fmGA optimizing the design process can be monitored in this module. In Result, the optimal sequence of performing design activities is provided. Users can also obtain the information of critical tasks and important information regarding the design process for management purpose.



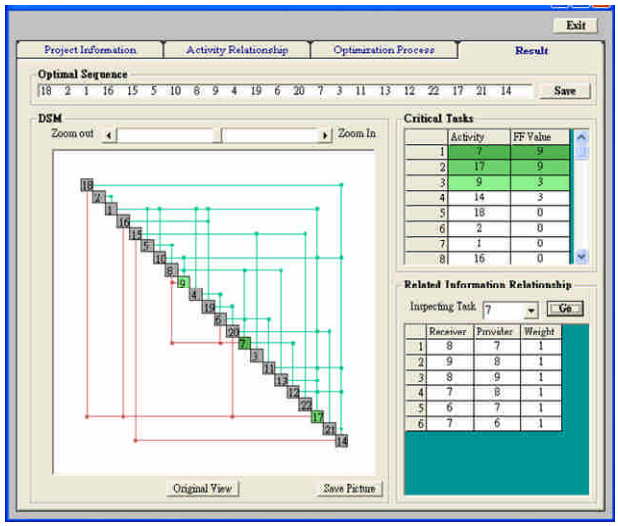
(a)



(b)



(c)



(d)

Figure 9. Interface of DPPO

A design process which contains ten activities is presented to verify the proposed model. Table 1 shows the information dependency and the information factors between activities of the design project under test. The test was conducted on the computer equipped with Pentium 4 3.2 GHz and 1G RAM. The result of this case study is shown in Figure 10. As Figure 10 depicts, the iterative loops within the design process is minimized. In addition, since the IFs provide the important of the information flow, the information flow with high weight is arranged as feedforward in stead of feedback. For example, the information flows from activity 3 to activity 1 and from activity to activity 7 are all arranged as the feedforwards since these two information flows has the highest weights.

Table 1 Information dependency and IF of the test example

Receiver	Provider	IF	Receiver	Provider	IF
A2	A1	0.3	A10	A6	0.6
A3	A1	0.2	A1	A7	1
A3	A2	0.2	A4	A7	0.5
A6	A2	0.8	A5	A8	0.5
A1	A3	1	A10	A8	0.9
A5	A3	0.4	A1	A9	0.3
A8	A3	0.6	A2	A9	0.3
A9	A4	0.6	A7	A9	0.4
A8	A5	0.5	A6	A10	0.6
A5	A6	0.7			

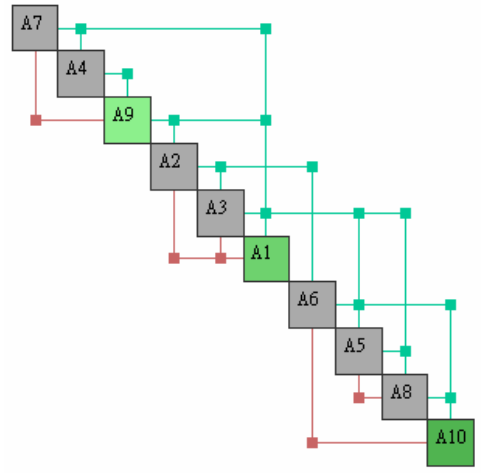


Figure 10. The result of the test example

6. CONCLUSIONS

This paper presents a model that integrate DSM and fmGA to the optimize design process for the large-scale design project. Three main contributions can be discovered from this study. First, the impact of the iterative loops within the design process can be clearly identified and reduced. Second, since the weight of each information flow can be identified, the information flows with high weights are arranged as feedforwards instead of feedbacks. Third, with the development of the computer implementation, project manager can effectively and efficiently optimize the design process.

ACKNOWLEDGEMENTS

This work was supported by the National Science Council, Taiwan under Grant No. NSC 94-2211-E-006-082.

REFERENCES

[1] Steward, D. V. (1981). *Systems analysis and management: Structure, strategy, and design.* Princeton, NJ: Petrocelli.

[2] Austin, S., Baldwin, A., and Newton, A (1996). "A data flow model to plan and manage the building design process." *Journal of Engineering Design*, 7(1), 3-25.

[3] Austin, S., Baldwin, A., Li, B., and Waskett, P.(1999). "Analytical Design Planning Technique: a Model of the Detailed Building Design Process." *Design Studies*, 20, 279-296.

[4] Austin, S., Baldwin, A., Li, B., and Waskett, P.(2000). "Analytical Design Planning Technique (ADePT): a Dependency Structure Matrix tool to Schedule the Building Design Process." *Construction Management and Economics*, 18, 173-182.

[5] Bloebaum, C. L. (1995). "Coupling Strength-based System Reduction for Complex Engineering Design." *Structural Optimization*, 10, 113-121.

[6] Browning T. R. (2001). "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions." *Transactions on Engineering Management*, IEEE, 48(3), 292-306.

[7] Chen, S. J., and Lin L. (2003). "Decomposition of interdependent task group for concurrent engineering." *Computers and Industrial Engineering*, 44, 435-459.

[8] Choo, H. J., Hamoond, J., Tommelein, I.D., Austin, S. A., and Ballard, G. (2004). "DePlan: a Tool for Integrated Design Management." *Automation in Construction*, 13, 313-326.

[9] Rogers, J. L. (1989). "DeMAID- a design manager's aide for intelligent decomposition user's guide." *NASA TM 101575*.

[10] Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. (1993). "Rapid Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms." *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56-64.

[11] Knjazew, D. (2003). OmeGA: A competent genetic algorithm for solving permutation and scheduling problems, Kluwer Academic Publishers, Boston, London.