A NEW APPROACH TO COMPONENT REUSE IN MULTI-SOFTWARE DEVELOPMENT PROJECT MANAGEMENT BY USING AN INFORMATION-CENTRIC PROJECT MODEL

Akihiro Sakaedani[1,2], Toshiyuki Yasui[3], Seiko Shirasaka[2] and Takashi Maeno[2]

[1]NTT Comware Corporation, Japan
[2]Graduate School of System Design and Management, Keio University, Japan
[3]Keio Advanced Research Center, Keio University, Japan

ABSTRACT

This paper evaluates the effects of component reuse by using a model that focuses on information transfer in software development projects. The cost models from earlier studies do not account for the difficulty of reuse, whereas the proposed information-centric model does consider the level of this difficulty. This paper has two main conclusions. First, component reuse does not always improve the productivity of a software development project. Second, component reuse is a valuable productivity tool if the following conditions are satisfied: (1) reuse results in simplification of project structures and (2) the level of difficulty of project elements is decreased from the reuse of components.

Keywords: software development, reuse, productivity, multi-project, information

1 REUSE MODEL OF SOFTWARE DEVELOPMENT

Software developers often recognize the advantage of component reuse in projects. Component reuse may increase productivity, but achieving effectiveness is difficult. According to Tracz (1995), lack of understanding on how to use components, difficulty in modifying components, difficulty in integrating/composing components, and other factors are obstacles to effective reuse.

The existing cost models do not reflect the difficulty of reuse adequately. In other words, the cost models cannot be said to adequately measure the effect of component reuse. A review of the existing cost models by Mili et al (2001) shows that none of these models consider the impact of know-how about reuse or the impact of successful work experience. In Mili's review, reuse models are classified in terms of cost factors (e.g., lines of source code, engineering costs, required investment to reuse, etc.). Models based on the number of lines of source code have been proposed by Schimsky (1992) and Poulin and Caruso (1993), but those models consider only source code length and thus cannot reflect the factor of knowledge transfer costs. Models based on engineering costs such as those of Bollinger and Pfleeger (1990) and Gaffney and Cruickshank (1992) do not account for project design, documentation, and work experience. Investment-based models such as the one proposed by Margano and Rhoades (1992) do not reflect the maintainability of the reused components or the ability to understand the reused components.

None of the previous cost models account for the information and know-how needed to reuse components effectively. These neglected factors coincide with obstacles that hinder the reuse of components.

2 PROBLEM

Lack of information is the main source of difficulty in reusing components (how to reuse components, modify reused components, integrate reused components, etc.). Thus, information transfer costs should be a cost factor in evaluating component reuse.

3 PURPOSE OF THIS STUDY

In this paper, the author evaluates the effect of component reuse by using a model that focuses on information transfer in project (Sakaedani et al., 2012; Sakaedani and Yasui, 2010). This model is based on a design structure matrix (DSM) and axiomatic design (Suh, 2001). This paper has two main objectives: (1) to investigate whether component reuse always improves the productivity of software development projects, and if not, (2) to explore the conditions under which component reuse is a valuable productivity tool in software development projects.

4 METHOD

Sakaedani et al (2010, 2012) developed a project model using a DSM. The project model focuses on information-carrying capacity. This study explains the negative relationship between information-carrying capacity and productivity. This information-centric project model is generally applicable to software development. However, the model has two restrictions. First, the model does not track changes over time and instead records a snapshot of the project. Second, the model does not explicitly show the sequence of tasks. However, according to Sakaedani et al (2012), the main work in software development projects is the transfer of information. Information to be transferred has two characteristics, information stickiness (von Hippel, 1994) and equivocality (Draft and Lengel, 1986), that can cause problems such as the repetition of work. Thus, this model considers these problems by using information-carrying capacity, rather than the sequence of tasks.

4.1 Concept

The project elements considered in this model are: teams, activities, artifacts, components, functions, requirements, features, and needs. Each element has relationships based on axiomatic design. In this concept, the metrics of information-carrying capacity account for the characteristics of information (information stickiness and equivocality). Figure 1 shows a conventional project model that considers information stickiness and equivocality. A large, complex project model is a complex mesh structure that is characterized by a high degree of equivocality among the elements of the project and a high degree of informational stickiness at many of the elements themselves. The result is low efficiency in the transmission of information. The idealized project model is a parallel structure. The project elements would have a low degree of equivocality and also be fairly simple (i.e., have a low degree of information stickiness). Under such an idealized model, the flow of information would be relatively smooth.

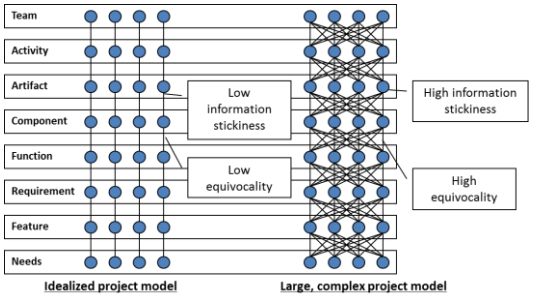

Figure 1. Project Model (from Sakaedani et al, 2010,2012)

This relationship defines the system matrix by applying the multiple domain matrix (MDM) method of Lindeman et al (2009).

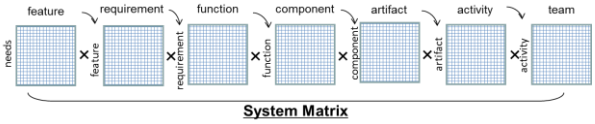

Figure 2. System Matrix (from Sakaedani et al, 2010,2012)

4.2 Formulation of evaluation indicators for the information-carrying capacity in a project model

4.2.1 Equivocality and interdependency

Sakaedani et al (2012) define equivocality as interdependency. Interdependency is the norm of system matrix **s** minus the unit matrix. The value of elements in this matrix is 0 or 1. A value of 0 means that

there is no relationship between the two elements. A value of 1 indicates that a relationship between the two elements exists. This equation measures the length of difference from the idealized project.

4.2.2 Information stickiness and difficulty

Sakaedani et al (2012) define information stickiness as difficulty. Thus, the level of difficulty can be quantified as information stickiness. The elements of the matrix are configured as the product of each difficulty. The difficulty is the norm of system matrix **n**.

4.2.3 information-carrying capacity and complexity

Information-carrying capacity affects the cost of information transfer. Sakaedani et al (2012) define information transfer cost as the product of equivocality and information stickiness. This cost is equal to the product of interdependency and difficulty. The complexity of the system matrix can be defined by the following equation:

$$1/\text{productivity} \propto \text{information transfer cost} \propto \text{complexity} \quad (1)$$

4.3 Strategy of productivity improvement

For productivity enhancement to occur, information transfer costs must be decreased. For a reduction in information transfer costs, complexity must be reduced. There are two approaches to reducing the complexity. The first is reducing interdependency. Thus, each DSM has to transition to diagonalization to the maximum extent. The second is reducing the level of difficulty. Thus, it is preferable that each element be simple or easy.

5 PRODUCTIVITY IMPROVEMENT FOR MULTI-PROJECTS

In multi-projects, another method exists to increase productivity: the reuse of design and implementation know-how (included components). However, project managers must evaluate the effect on information-carrying capacity from the reuse of components. A decline in information-carrying capacity results in lower productivity. This study further develops this system matrix and the new model can measure the effect of reuse.

5.1 Model for evaluating the reuse of design information in multiple projects

The matrix consists of several submatrices, where the system matrices of the projects are located in the diagonal positions. Other submatrices indicate the relationship between the reused elements.

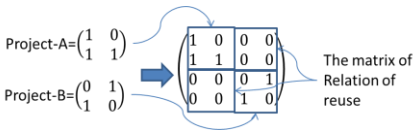

Figure 3. Matrix of reuse relationship

6 TRIAL CASE

The trial model is shown below in Figure 4.

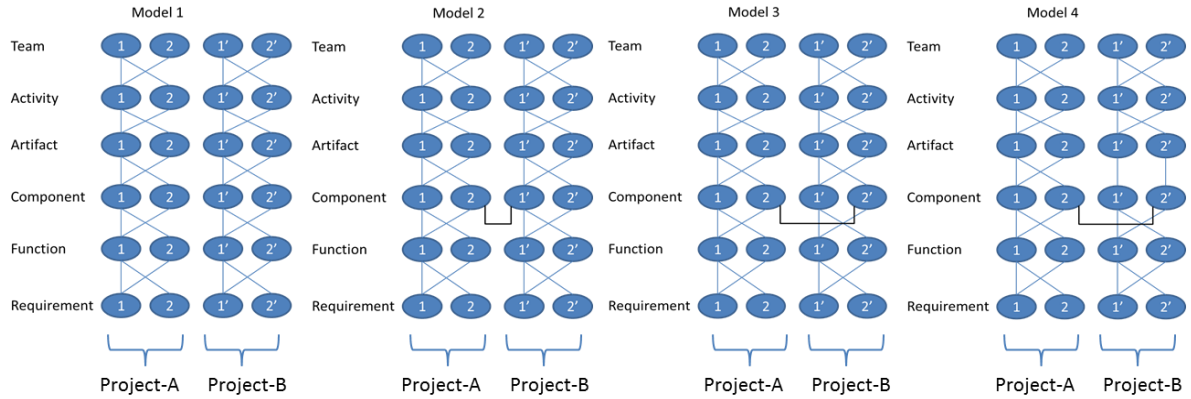

Figure 4. Trial case model

Model 1 shows two independent projects. Models 2, 3, and 4 show projects with reused components. In Model 2, the project reuses component 2 and 1'. In Models 3 and 4, the projects reuse component 2 and 2'. Model 4 is a modified version of Model 3, in which the relationship between artifacts and

components is simplified to 1 to 1.

6.1 Evaluation of complexity

Each model is transformed into system matrix (Figure5). These matrixes are system matrixes **s**. In this case, all elements of system matrix **n** are 0 or 1. Thus system matrixes **n** are the same as system matrixes **s**.

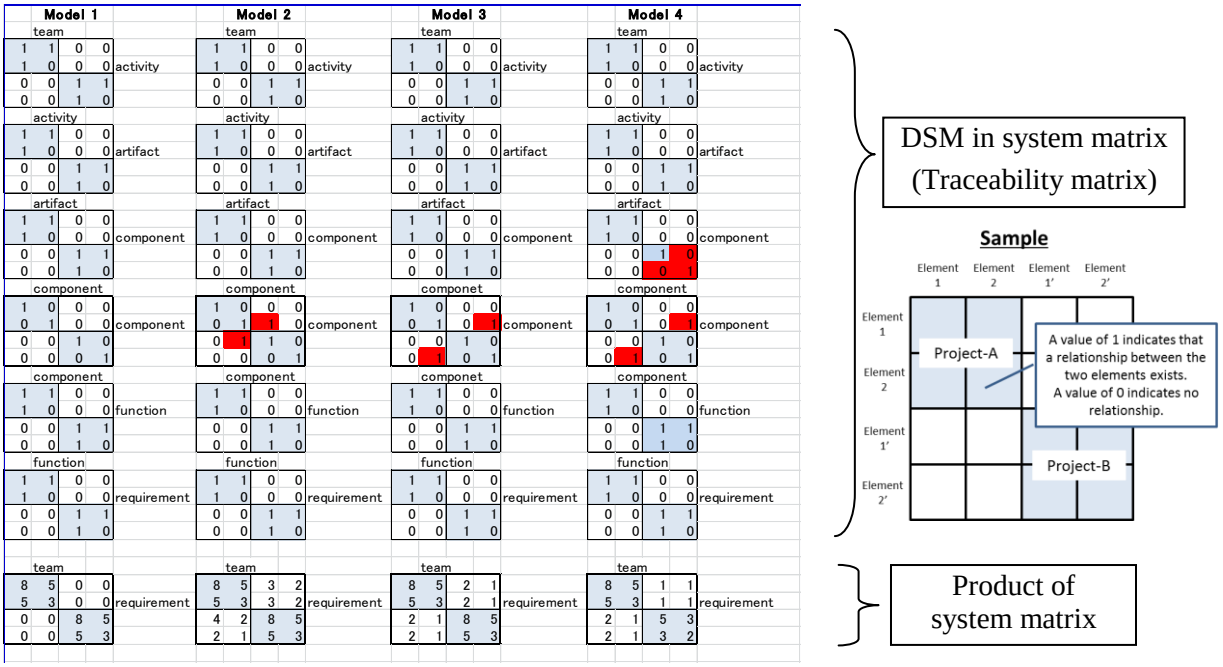

Figure 5. System Matrix

The difficulty is the norm of product of system matrix **n**, interdependency is the norm of system matrix **s** minus the unit matrix. Complexity is equal to the product of interdependency and difficulty. As a result, Model 1, which has no reused components, is less complex than Model 2. However, Model 2 is more complex than Model 3, which has reused component. Model 4 is less complex than Model 3.

Table1 Complexity

	Model 1	Model 2	Model 3	Model 4
Difficulty	15.68	17.23	16.31	13.56
Interdependency	14.35	16.03	15.03	12.33
Complexity	225	276	245	167

7 DISCUSSION

7.1 Model 1 and Model 2

Component reuse does not always reduce complexity and increased complexity can result in a decline in productivity. Thus, component reuse does not always reduce productivity. As can be seen, Model 1 does not reuse components, but both Model 2 and Model 3 do. Compared with Model 2, Model 1 is less complex. Components reuse alone increases the complexity, which in this case results in a decline in productivity.

7.2 Model 3 and Model 4

The structure of project elements, excluding the reuse of components, affects the efficiency achieved at a given level of complexity. Simply put, the structure of a project has an effect on reuse. This result goes hand in hand with earlier studies.

7.2.1 Interdependency in projects

The structure of projects has an impact on the effect of component reuse. Simplification of project structure enhances the effectiveness of components reuse.

This reason is that the interdependency of the artifact and component elements is different between Model 3 and Model 4. In Model 4, one matrix is a unit matrix, which decreases the difficulty, the interdependency, and the complexity. Thus, the simplification of project structure enhances the effectiveness of components reuse.

7.2.2 Difficulty of the elements in projects

All difficulty is equal to 1 in this trial case. However, this section is devoted to verifying that change in difficulty has an impact on complexity.

For example, the difficulty is equal to 1.5 for reused components and the difficulty of artifact, activity that is related to the reused components, is equal to 1.5 in Model 4. As a result, the complexity is equal to 264. This complexity is higher than Model 1. Instead, if the level of difficulty declines from component reuse, the complexity will decline.

7.2.3 Generalization of the reuse model

This section attempts to generalize the concepts above. Equations (2) and (3) below define the system matrix, where Matrix A is a no-reuse model and Matrix B is a reuse model. Each element $\mathbf{a_i}$, $\mathbf{b_i}$, and $\mathbf{c_i}$ is a submatrix. The submatrices $\mathbf{a_i}$ and $\mathbf{b_i}$ are on diagonal in the system matrix.

$$A = \begin{pmatrix} a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_n \end{pmatrix} \qquad B = \begin{pmatrix} b_1 & \cdots & c_1 \\ \vdots & \ddots & \vdots \\ c_n & \cdots & b_n \end{pmatrix} \tag{2}$$
$$a_i \geq 0 \qquad b_i \geq 0, c_i \geq 0$$

Based on these models, the effect of reuse is organized by the structure of the project (Table 2).

Table 2. The effect of reuse and the structure of the project

	Structure	Interdependency	Difficulty	Complexity	Effect of reuse
1	More complex of reuse project	m<k	n<l	complexity(a)<complexity(b)	No effect
2	More interdependent and less difficulty, or less interdependent and more difficulty	m<k	n>l	complexity(a)≤complexity(b) or complexity(a)≥complexity(b)	Either way
3		m>k	n<l		
4	More simply of reuse project	m≥k	n≥l	complexity(a)≥complexity(b)	Effective

Equation (3) is defined as the following:

$$complexity(a) = \|A - E\|\|A'\| \qquad complexity(b) = \|B - E\|\|B'\| \tag{3}$$
A= system matrix-**s** B= system matrix-**s**
A′= system matrix-**n** B′= system matrix-**n**
$$m = \|A - E\| \quad n = \|A'\| \qquad k = \|B - E\| \quad l = \|B'\|$$

If **m** is less than **k** and **n** is less than **l**, then reuse is not effective. If **m** is more than **k** and **n** is more than **l**, the reuse has an effect. If **m** is less than **k** and **n** is more than **l** or if **m** is more than **k** and **n** is less than **l**, the results are ambiguous and reuse does not always work. In other word, if reuse of components results in the project structure becoming simpler and the level of difficulty decreasing, then the project will have increased productivity.

8 CONCLUSION

According to the model focusing on information transfer in project, component reuse does not always improve the productivity of software development projects. The cost models from earlier studies do not take into account the difficulty of reuse, whereas this information-centric model does consider the level of difficulty in reusing components. If the following two conditions are satisfied, then reuse can act as a valuable productivity tool. First, reuse must simplify the project structure. Second, the level of difficulty of the elements related to the reused components must decrease.

9 ACKNOWLEDGMENT

This study was financially supported by the Center for Education and Research of Symbiotic, Safe and Secure Systems Design at the Keio University Advanced Research Center through the Ministry of Education, Culture, Sports, Science and Technology (MEXT) Global COE Program (Keio University GCOE H-10), as well as by a MEXT Grant-in-Aid for Scientific Research, 2011 Science Research(c), "Application Research of System Design Methodology for Integration of Social System and Technology System" (No. 23611038).

REFERENCES

Bollinger, T., and S. Pfleeger. (1990). Economic of Reuse: Issues and Alternatives. Information and Software Technology, December, 32(10): 643-652.

Draft, R. l., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. Management Science, 32(5), 554-571.

Gaffney, J.E. and R.D. Cruickschank (1992), "A General Economics Model of Software Reuse," In Proceedings of the Internat. Conf. on Software Engineering, Melbourne, Australia, May, pp. 327–337.

Lindeman, U., Maurer, M., Braun, T. (2009). Structural Complexity Management An Approach for the Field of Product Design. : Springer.

Margano, J. and T.E. Rhoads (1992), "Software Reuse Economics: Cost Benefit Analysis on a Large Scale Ada Project," In Proceedings of Internat. Conf. on Software Engineering, Melbourne, Australia, May, pp. 338–348.

Mili, A., Fowler Chmiel, S., Gottumukkala, R., and Zhang, L. (2001). Managing Software Reuse Economics: An Integrated ROI-based Model, Annals of Software Engineering 11,175-218.

Poulin, J.S. and J.M. Caruso (1993), "A Reuse Metrics and Return on Investment Model," In Advances in Software Reuse: Proceedings of the 2nd Internat. Workshop on Software Reusability, Lucca, Italy, March, pp. 152–166.

Sakaedani, A.,Yasui, T.(2010). "Inter-Element Relations in the Configured Systems: Second Dimension of the System Complexity from the Case Study of the Japan's ANIME Industry" Proceedings of 4th Asia-Pacific Conference on System Engineering

Sakaedani, A., Ohkami, Y., Kohtake, N. (2012). Construction of a traceability matrix for high quality project management. Synthesiology, 5(1), (in press).

Schimsky, D. (1992). Software Reuse –Some Realities. Vitro Tech Journal, Summer, 10(1): 47-57.

Suh, N. P. (2001). Axiomatic Design: Advances and Applications. Oxford University Press, New York.

Tracz, W.(1995). Confessions of a Used Program Salesman Institutionalizing Software Reuse, New York: Addison-Wesley Publishing Company.

von Hippel, E. (1994). "Sticky information" and the locus of the problem solving: implications for innovation. Management Science, 40(4), 429–439.

Contact: Akihiro Sakaedani

Graduate School of System Design and Management, Keio University

Core Technology, Quality Management and Engineering Division, NTT Comware Corporation

NTT Shinagawa TWINS Annex Bldg. 22F

1-9-1 Konan, Minato-ku, Tokyo 108-8019

Japan

Phone: +81.3.5796.4886

e-mail: a.sakaedani@nttcom.co.jp, a.sakaedani@sdm.keio.ac.jp