Paper #1569319363

Complexity Management by Using Project Matrix

Akihiro Sakaedani[1)], Toshiyuki Yasui[2)]

1) NTT COMWARE CORPORATION / Fellow, the SDM Institute Keio University
NTT Makuhari Bldg 21F, 1-6 Nakase, Mihama-ku, Chiba 261-0023 Japan
a.sakaedani@nttcom.co.jp
2) Keio University Graduate School of System Design and Management
Kyoseikan, 6F - 4-1-1 Hiyoshi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8526 Japan
t.yasui@z2.keio.jp

Abstract

The more complexity a software projects contain, the more frequently the developments fail. However we can reduce the systems complexity by using a matrix which I propose in this paper. A software development project could be divided into 8 elements, and the complexity of the system development depends on the interdependency and complex relations between element and one another. This paper explains how project complexity is well managed by the interdependency and simple relationship.

1. Introduction

Technology progressions are not solution of software development. Projects might be more complex. And failure rates of system developments do not decrease by only improving development process. Thus we should consider another dimension of the systems complexity which leads us to new approach to improve the success rates of complex system development.

2. Structure of software development project and its problem

The project contains 8 elements: teams, activities, artifacts, components, functions, requirements, features and needs. If these elements are independent, it will be easy to manage projects. The complexity of a project is defined in terms of a difficult level of element itself and interdependency between element and other elements. The difficulty level of element itself tends to reduce by the progression of technology. But, even if the difficulty level would be reduced, if the interdependency will be still complex, the total effectiveness could be offset. In other words, the result might be same or more complexity. Basically, it is important to not only reduce the difficulty level of element itself but also simplify the relations between elements. (Figure1)

It is a challenge to realize it that we visualize the project totally and well organize the relations between elements.

Difficulty rate of elements	dependency rate of relations	Complexity of Project
Large increase	Small reduction	Increase
Small increase	Large reduction	Reduction
Small reduction	Large increase	Increase
Large reduction	Small increase	Reduction

Figure1 Difficulty and Independency

3. The concept of proposed model

This paper shows a model that can be useful if it will be used to analyze complexity of elements in a project qualitatively and quantitatively. It will help us to well understand the complexity of software development projects. As I mentioned before, eight elements of a system development project are “Needs”, “Features”, “Requirements”, “Functions”, “Components”, “Artifacts”, “Activities” and “Teams”. I think

those elements have relationships like this:

At the beginning, “Needs” are broken down into “Features”. Next, “Features” are defined as “Requirements”. “Functions” realize the “Requirements”. “Components” implement the “Functions”. In the design of the “Components”, it is necessary to make “Artifacts”. At the end, “Teams (organizations)” execute “Activities” in the process of making the “Artifacts”. Then, the design information is transmitted to these eight elements.

If an element connects to just another, it is simple relation (Figure2, (left)). In this case, 8 elements of a project could be connecting as one line from “Team” to “Requirement”. So, it is one line one project. Thus, each line of a project has not interdependency with other lines. So, the line of a project is able to execute only the project. But, in many cases, a project of software development has a lot of connections to elements in other lines and the difficulty level of elements is also high (Figuer2, (right)).Therefore, design work and elements could be interdependency and affect each other. And if the difficulty level of elements is high, it might lose the design information. Like this, in a development project, the elements in a project should have single relation with other element which is contained in the project. The complexity of relation is reducing the quality of a design work itself.

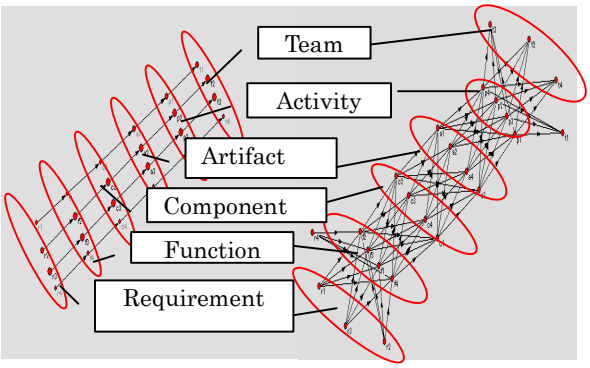


Figure2 Elements and the relationships

4. How to conceptualize and use model

4.1. Concept

If the network diagram (Figure2) is transformed into a matrix (Figure 3), the matrix model can be used to perform quantitative analysis. The matrix can organize eight elements.

Additionally, for example, a project manager needs administrative documents for managing artifacts. This matrix model can be made easily by using an artifact-artifact matrix (bottom five matrixes in Figure 3).

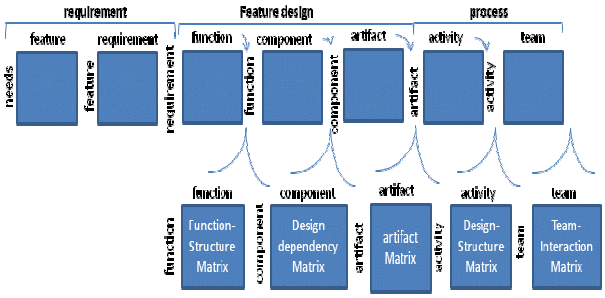


Figure 3 Matrix Model

4.2. How to set the each values in the matrix

In this section, I will explain how to set the each value in the matrix. If there is no relationship between elements, the value will set to “0”. If there is a relation between elements, the value will set to “1” (Figure 4).

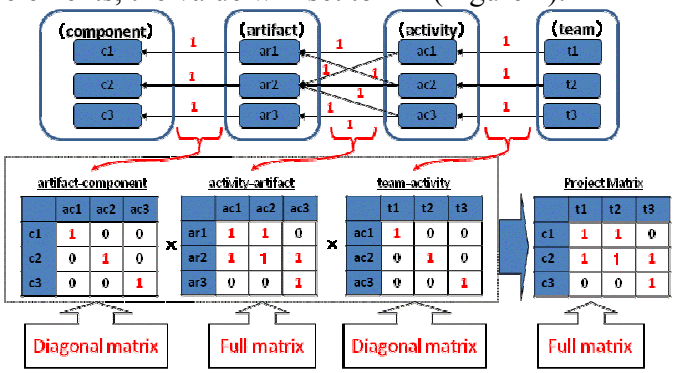


Figure4 How to set the elements value

For example, if there is strong relation, the value will not be set to “1” but set to “0.5”. The number is selected by project members. It is important for a project to define the numeric standards beforehand.

A diagonal matrix is the most independent of the relation. In other words, design information adequately reflects each element. A triangular matrix is more dependent than dose a diagonal matrix. A full matrix is the most dependency matrix.

4.3. Qualitative analysis of a matrix

If the matrix model consists of a full matrix, the project will be complexity. As shown in Figure 5, in case of including one or two full matrixes, all results of multiplying matrices is a full matrix.

$$[\backslash][X][\backslash] = [X] \rightarrow ①$$
$$[X][\backslash][X] = [X] \rightarrow ②$$
$$[\backslash][LT][X] = [LT][X] = [X] \rightarrow ③$$
$$[X][LT][\backslash] = [X][LT] = [X] \rightarrow ④$$
$$[\backslash][\backslash][\backslash] = [\backslash] \rightarrow ⑤$$
$$[X][X][X] = [X] \rightarrow ⑥$$
$$[LT][LT][LT] = [LT] \rightarrow ⑦$$
$$[\backslash][LT][LT] = [LT] \rightarrow ③'$$
$$[LT][LT][\backslash] = [LT] \rightarrow ④'$$

$[\backslash]$ =diagonal matrix
$[LT]$ =triangle matrix
$[X]$ -full matrix

Figure5 Matrix Analysis

In this figure, it is easy for us to see which pattern is the simplest one. The simplest case is the pattern 5. In another case, the results are full or triangular matrixes. In other words, the project will be complex. The full matrix is root caouse of complexity. It is important for project menbers to focus on managing the relations with a full matrix.

4.4. Quantitative Analysis by Quality, Cost, Delivery

In order to understand the project, we analyze the result of multiplied matrixes.

① Project Quality (Complexity)

The project complexity is defined by both the interdependency and the difficulty of the configuration elements.

The interdependency is determined by an inner product (figure6). This concept is similar to "Vector space model" ("term vector model").

The difficulty is caused by the value of the configuration elements. In this study, the difficulty is the norm of project matrix.

As a result, project complexity is defined as

Project complexity =cosθ×1／norm

Consider the following example.

(r1, r2) are vectors pointing in the requirement, (t1, t2) are that vectors pointing in the team.

$$\begin{pmatrix} r1 \\ r2 \end{pmatrix} = \begin{pmatrix} \text{Requirement-} \\ \text{Function} \end{pmatrix} \times \begin{pmatrix} \text{Function-} \\ \text{Component} \end{pmatrix} \times \begin{pmatrix} \text{Component-} \\ \text{Artifact} \end{pmatrix} \times \begin{pmatrix} \text{Artifact-} \\ \text{Activity} \end{pmatrix} \times \begin{pmatrix} \text{Activity-} \\ \text{Team} \end{pmatrix} \times \begin{pmatrix} t1 \\ t2 \end{pmatrix}$$
$$= \begin{pmatrix} 1/2 & 1 \\ 1 & 1/2 \end{pmatrix} \times \begin{pmatrix} t1 \\ t2 \end{pmatrix}$$

To evaluate the project, we calculate the eigenvalues, and eigenvectors.

Eigen values are λ=3/2、-1/2

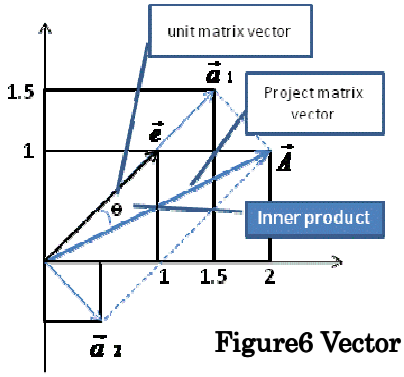


Figure6 Vector Space

The norm is the maximum eigenvalue = |3/2| = 3/2. The difficulty is calculated by taking the inverse and equal 2/3.

Further, if λ = 3/2 and eigenvector to calculate eigen vector a1 = (1, 1),

If λ = -1/2, eigenunique vector a2 = (- 1, 1)

As a result, A = 3/2 × a1 + (-1/2) ×a2 = (2, 1)

$$\vec{A} \cdot \vec{e} = (2,1) \cdot (1,1) = 2 \times 1 + 1 \times 1 = 3$$
$$\vec{A} \cdot \vec{e} = |\vec{A}||\vec{e}| \cos\theta$$
$$\cos\theta = 3/\sqrt{10} \fallingdotseq 0.949$$

As a result the interdependency is 0.949.

The project complexity is multiply interdependency to difficulty.

2/3×0.949=0.632 0.63

② Project Cost

When the requirement vector (y) has the unit of "¥" and the team vector (x) has the unit of "manpower", matrix A shows just "Yen/person". Matrix A is expressed in "y = Ax". Therefore, it is possible to derive the theoretical price of each requirement.

$$\begin{pmatrix} r1 \\ r2 \\ r3 \\ r4 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}}_{\text{Requirement-Function}} \times \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}}_{\text{Function-Component}} \times \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}}_{\text{Component-Artifact}} \times \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}}_{\text{Artifact-Activity}} \times \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}}_{\text{Activity-Team}} \times \begin{pmatrix} t1 \\ t2 \\ t3 \\ t4 \end{pmatrix}$$

$$= \begin{pmatrix} 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \end{pmatrix} \times \begin{pmatrix} t1 \\ t2 \\ t3 \\ t4 \end{pmatrix}$$

In this case, each elements costs 2560000 Yen/person, and the team vector is (t1, t2, t3, t4) = (4 month, 2 months, 2 month, 1 month), we have

$$= \begin{pmatrix} 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \\ 256 & 256 & 256 & 256 \end{pmatrix} \times \begin{pmatrix} 4 \\ 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2304 \\ 2304 \\ 2304 \\ 2304 \end{pmatrix}$$

The requirement vector (r1, r2, r3, r4) can be calculated as (23040000 yen, 23040000 yen, 23040000 yen, 23040000 yen).

③ Project Delivery

The delivery process depends on two matrices. Two matrices are team-activity and activity-artifact.

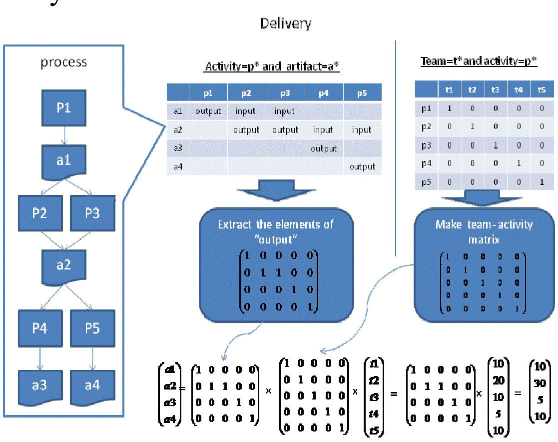


Figure7 Concept of Delivery

As shown in Figure7,

Step1: We prepare the activity-artifact and team-activity matrixes.

Step2: In matrix of activity-artifact, we focus on the output columns and then rebuild the matrix.

Step3: We set the values in the two matrixes. The Delivery matrix is created by multiplying the two matrixes (activity-artifact, team-activity).

If each team requires the work days (t1, t2, t3, t4, t5) = (10, 20, 10, 5, 10) days, artifacts (a1, a2, a3, a4) will be equaled to (10, 30, 5, 10). In this example, it is possible to calculate the delivery period, because the entire process is clear. Activities p4 and p5 are parallel tasks. Therefore, when artifact a4 exits, the task is complete. As a result, the delivery period is this calculation is, a1+a2 +a4 = 10 + 30 + 10 = 50 days.

5. Verification by case study

As shown in Figure 8, this is the sample cases of a software development project

5.1. Use case model

In this use case study, there are two actors, a customer and a bank.

The “Log in” use case has three requirements.

Req1. System can confirm user.

Req2. System can change the password (PW).

Req3. System can lock out the users, if they will enter a wrong password for consecutively three times.

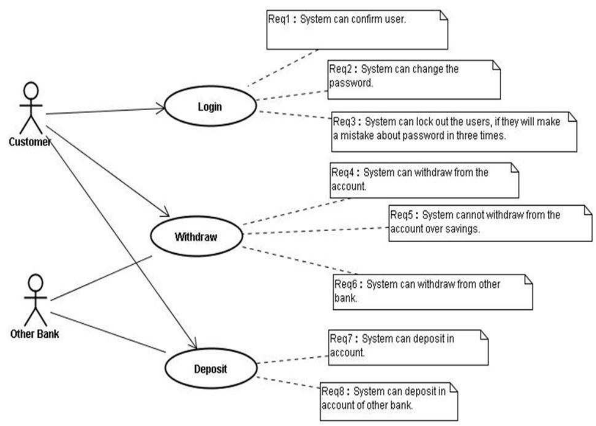


Figure 8 Use Case Model

The "withdraw" use case has three requirements.

Req4. System can withdraw from an account.

Req5. System cannot withdraw from an account over savings.

Req6. System can withdraw from another bank.

The “Deposit” use case has two requirements.

Req7. System can deposit in account.

Req8. System can deposit in account of other bank.

This case study verifies the matrix model. In addition, (1) if you deploy a source code generator, (2) if you use package (PKG) software and (3) if you prepare source code by yourself(”scratch”). These three cases are compared and verified. Their network diagrams are shown as additional illustrations in “Appendixes 1, 2, and 3”. These figures use the UML class diagram. The stereotype is “matrix category”. The teams are project management, requirement management, architect, detail design, implementation, and test. The activities are business logic design, DB (database) design, and the user interface design. The artifacts are user interface specification, user interface transition diagrams, work flow diagram, feature specifications, system interface specifications, entity-relationship diagram, and source code. The components are the login screen, menu screen, withdrawing screen, on site screen, administration, withdrawing, savings, bank-bank interface, customer management data, deposit management data. The functions are the user ID (UID) input features, password (PW) input features, menu selection feature, withdraw amount input features, deposit amount input features, UID-PW management features, deposit balance management feature, bank-bank interface features, user search feature, registration withdrawing amount, and registration deposit amount. These functions realize the requirements.

5.2. Comparative study(cost, productivity)

In this section, we have tested the assumption that each team vector has “one” manpower (effort). Teams have five efforts in total. If a matrix means productivity, the requirement vector will be the price of each request. (Figure9)

(1) Generator: five months of effort, work of 2000000 yen

(2) PKG: five months of effort, work of 1800000 yen

(3) Scratch: five months of effort, work of 2400000 Yen

As a result, the productivity is less 360000 / man month (PKG) than 480000 / man month (scratch). PKG is the best in terms of productivity. Scratch development has the poorest productivity.

	Requirement								
	System can confirm user.	System can change the password (PW).	System can lock out the users, if they will make a mistake about password in three times.	System can withdraw from the account.	System cannot withdraw from the account over savings.	System can withdraw from other bank.	System can deposit in account.	System can deposit in account of other bank.	Total
Generator	35	20	20	45	35	5	35	5	200
PKG	25	20	20	35	35	5	35	5	180
Scratch	45	25	25	55	40	5	40	5	240

Figure 9 Cost Comparison

In this case study, all elements has been set the value 1. Normally, the elements value must be decided according to the actual efforts.

5.3. Comparative study (quality)

In this section, it is shown complexity in Figure 10. Interdependency is calculated from cosθ(inner product). The unit matrix is the baseline (quality value = 1). Scratch development has the lowest value. “Scratch” is the most dependent on each element, and the most difficult one in these cases. In a network-related diagram, the scratch case has the most complex relation. The following is a generator.

	Indopendency = cos θ	Difficulty = 1/Norm	Complexity = Independency × Difficulty
Generator	0.873704	1/36.193922	0.024140
PKG	0.936796	1/32.093613	0.027632
Scratch	0.868290	1/43.703547	0.019868

Figure 10 Comparative Study of complexity

5.4. Improvement

In three appendixes, the teams are related with many activities. The requirements are realized by many functions. These are the points of improvement. Project members need to discuss about the structure. They must consider how to decrease the number of relations between the teams and their activities,

and between the requirements and the corresponding functions. This model helps visualize project.

6. Examination

6.1. To-Be model

It is possible to describe the structure of a project by the model proposed in this paper.

In software engineering, the each elements of a software development project have been discussed such as process, tools, and requirements. However the relations of each element have not been discussed. The proposed model gives you a bird's-eye view of the project structure. Further, you can obtain the quantitative status of project. We find that the most desirable configuration of a project is shown on a unit matrix.

6.2. Quantitative evaluation of productivity

In this research, it is possible to evaluate the project quality (complexity) by the theoretical calculations. For example, the approach using a generator is more productive than scratch development. According to calculation, productivity is expected to increase 17%. In real development project, using a generator, we have measured productivity increases 18.4%. This calculation result is a sufficient proof of the feasibility of the proposed model.

6.3. Analyzing about Complexity

It is possible to quantitatively and qualitatively analyze the project. Diagonal matrix is more productive than triangle matrix. And Triangle matrix is more productive than full matrixes. After all, the matrix which satisfies with Axiomatic Design (independence axiom and information axiom) is the most productive. In the case of your project, it is necessary for you to carefully consider the develop strategy about the relations of elements. Complexity determines the robustness of the project. No matter how the project members implement technology solutions, it will be disordered by a wrong strategy.

6.4. Complement the blind spot in CMMI

CMMI (capability maturity model integration) is useful to evaluate the organization's activity. But this model is not to mention the other elements (components function, etc). The methodology described in this research evaluates all elements of project (team, activity, artifact, component, function, and requirement).

7. Conclusion

This paper proposed a matrix model that represents the relationships between project elements.

An ideal model is unit matrix. By comparing each matrix to this criteria (unit matrix), this model will helps understand structure. As a result, larger, more complex software project will be successful. This model will allow software to continue providing reliable service as a social infrastructure.

8. Future Study

In this paper, interdependency is determined by the "inner product". However, in future study, we will attempt to calculate interdependency by using the "norm". The "norm" is the length as between a project matrix and a unit matrix.

References

Leffigerwell, D. and Widrig D.,. *Managing Software Requirements A Unified Approach.* : Addoison-Wesley, 2000.

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

Sakaedani, A., "QCD improvement by practical use of a standard Framework" *24th Symposium on Quality Control of Software, JUSE,* 2005.

Sosa, M.E.. "A Network Approach to Define Modularity of Components in Complex Products", *ASME Journal of Mechanical Design, vol129, no11, pp1118-1129, November 2007.*

"Component Modularity, Team Network Structure, and the Attendance to Technical Interdependences in Complex Product Development". *INSEAD Faculty & Reseach Working paper, 2006.*

"Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions".. *ASME Journal of Mechanical Design, vol125, no2, pp240-252,June 2003.*

Suh, N.P., *Axiomatic design*- Oxford University Press, USA, 2001

Biography

Akihiro Sakaedani

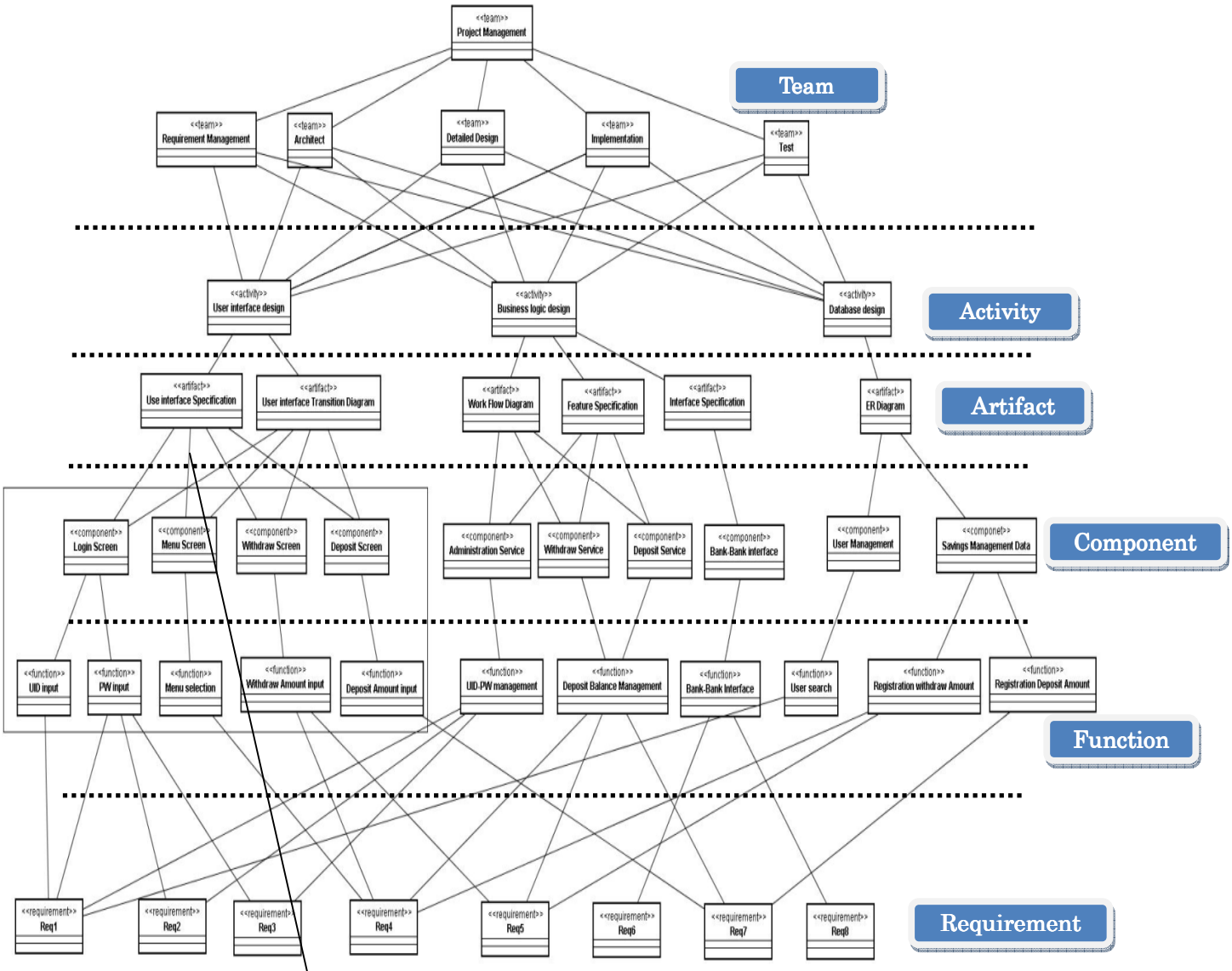
Upon graduating from the Gakushuin University (MS in Physics) in 1994, Akihiro Sakaedani joined NTT (Nippon Telephone and Telegram). He has been working for as a system engineer since about last 15 years. He studied in Keio University Graduate School of System Design and Management (MS in System Design and Management) (2008-2009). He is Fellow of the SDM Institute.

Toshiyuki Yasui

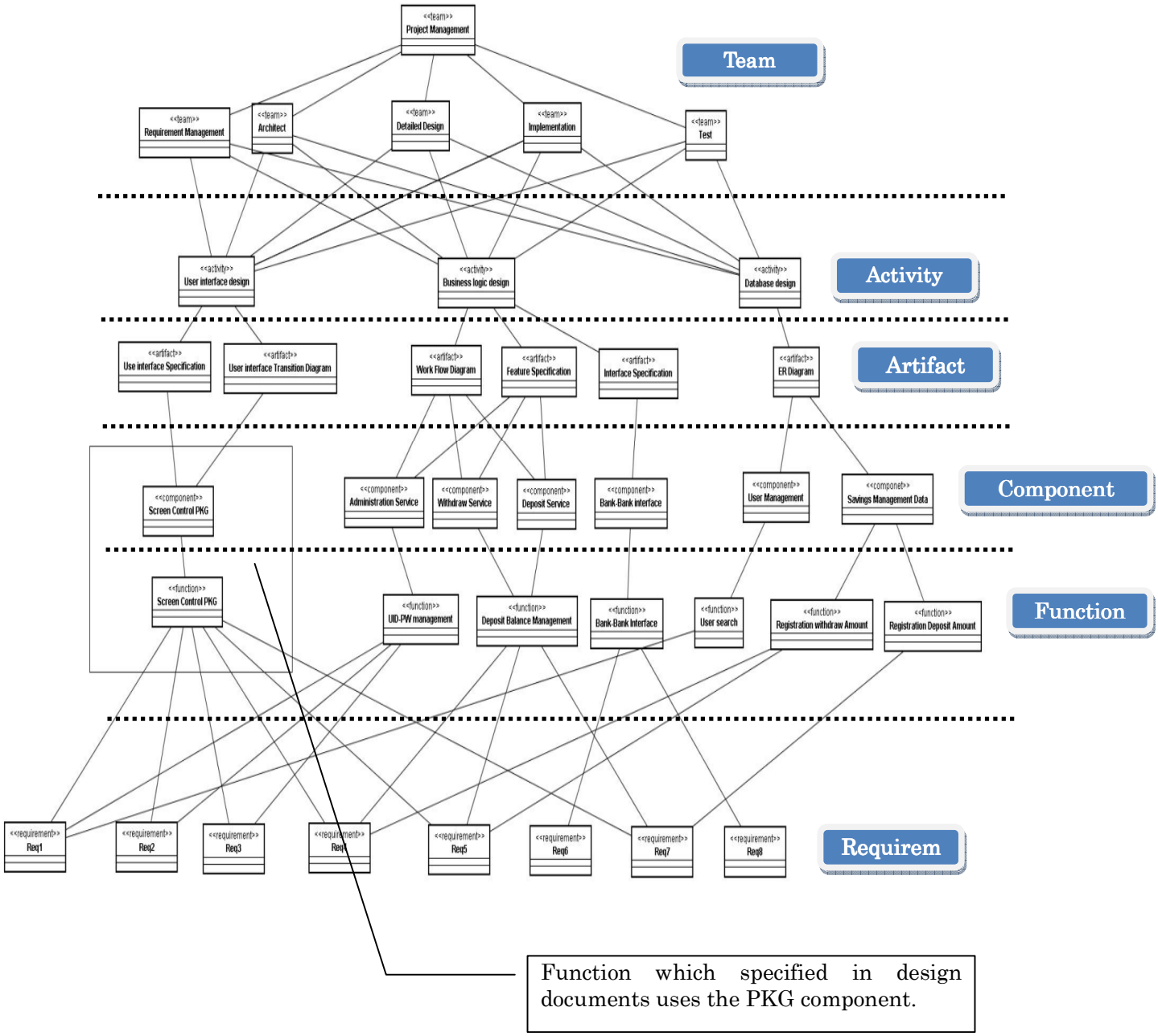
Special Research Professor, Graduate School of SDM, Keio University

Upon graduating from the University of Tokyo (BA in International Relations) in 1985, Toshiyuki Yasui joined the Japanese Ministry of Finance. He spent 25 years in various government posts including Councilor for the Financial Services Agency (2008-2009), Director for the FILP Research & Planning Office of the Ministry of Finance (2000-2001), and First Secretary in the Embassy of Japan for India (1992-1995). He also worked as Senior Fellow of the Institute of JBIC (2001-2005) and Trainee-Consultant for the OECD (1987-1989). He is a former Visiting Professor of Chuo University (2007-2008).

Appendix 1 In case of using a Generator



Appendix2 In case of using Package Software



Appendix3 In case of Scratch