A Process Complexity-Product Quality (PCPQ) Model Based on Process Fragment with Workflow Management Tables

Masaki Obana[1], Noriko Hanakawa[2], and Hajimu Iida[1]

[1] Nara Institute science and Technology,630-0192
8916-5 Takayama-cho, Ikoma, Nara Japan
{masaki-o,iida}@is.naist.jp
[2] Hannan University,Information management,580-8502
5-4-33, Amami-Higashi, Matsubara, Oasaka Japan
hanakawa@hannan-u.ac.jp

Abstract. In software development projects, large gaps between planned development process and actual development exist. A planned process is often gradually transformed into complicated processes including a base process and many process fragments. Therefore, we propose a metric of process complexity based on process fragments. Process fragments mean additional and piecemeal processes that are added on the way of a project. The process complexity depends on three elements; the number of group of developers, the number of simultaneous process, and ratio of an executing period for a period of the whole project. The process complexity was applied to six industrial projects. As a result, changes of process complexities in the six projects were clarified. In addition, we propose a procedure of making a PCPQ (Process Complexity-Product quality) model that can predict post-release product quality on the way of a project. As a result of making a PCPQ model using the six projects, a post-release product quality was able to be predicted.

1 Introduction

In software development projects, large gaps between planned development processes and actual executed development processes exist[1]. Even if a development team has originally selected a waterfall model unplanned small processes are often triggered as shown the following examples.

i. One activity of waterfall-based process is changed into new iterative process because of urgent specification changes.
ii. Developers design GUI using a new prototype development process at design phase.
iii. In an incremental development process, developers correct defects in the previous release while developers are implementing new functions in current release.

That is, the waterfall-based process at planning time is often gradually transformed into the combination of the original waterfall-based process and several unplanned

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 171–185, 2011.

small processes (hereinafter referred to as process fragments). In consequence, actual development processes are more complicated than planned one (see also Figure 1).

In this paper, we firstly assume that complicated process decreases product quality, and then propose a new metric for process complexity based on the number of unplanned process fragments, the number of simultaneous execution processes, and the number of developers' groups. It can be used to visualize how process complexity increases as actual development process proceeds.

An aim of measuring process complexity is to prevent software from becoming low quality product. A process complexity is derived from a base process and process fragments. A base process means an original process that was planned at the beginning of a project. Process fragments mean additional and piecemeal processes that are added to the base process on the way of a project. Process fragment occurs by urgent changes of customers' requirement, or sudden occurrence of debugging faults. Process fragments can be extracted from actual workflow management tables which are popular in Japanese industrial projects [2]. We especially focus on simultaneous execution of multiple processes to model the process complexity. Simultaneous execution of multiple processes is caused by adding process fragments on the way of a project. Finally, we propose a "process complexity – product quality" (PCPQ) model for predicting final product quality. We perform an industrial case study in order to show the usefulness of PCPQ model. In this case study, we found that PCPQ model was able to indicate the degree of post-release faults.

Section 2 shows related work about process metrics, and risk management. The process complexity is proposed in section 3. Section 3 also describes how the proposed complexity would be used in industrial projects. Case studies of six projects are shown in section 4. In section 5, we propose a PCPQ model to predict post-release product quality. Summary and future works are described in Section 6.

2 Related Work

Many software development process measurement techniques have been proposed. CMM [3] is a process maturity model by Humphrey. Five maturity levels of organizations have been proposed in CMM. When a maturity level is determined, various values of parameters (faults rate in total test, test density, review density) are collected. In addition, Sakamoto et al. proposed a metrics for measuring process improvement levels [4]. The metrics were applied to a project based on a waterfall process model. Theses process measurement metrics' parameters include the number of times of review execution and the number of faults in the reviewed documents. The aim of these process measurement techniques is improvement of process maturity of an organization, while our research aims to measure process complexity of a project, not organization. Especially, changes of process complexity in a project clearly are presented by our process complexity.

Many researches of process modeling techniques have been ever proposed. Cugola et al. proposed a process modeling language that describes easily additional tasks [5]. Extra tasks are easily added to a normal development process model using the modeling language. Fuggetta et al. proposed investigated problems about software development environments and tools oriented on various process models [6]. These process

modeling techniques are useful to simulate process models in order to manage projects. However, these process modeling techniques make no mention of process complexity in a project.

In a field of industrial practices, Rational Unified Process (RUP) has been proposed [7]. The RUP has evolved in integrating several practical development processes. The RUP includes Spiral process model, use-case oriented process model, and risk oriented process model. Moreover, the RUP can correspond to the latest development techniques such as agile software development, and .NET framework development. Although problems of management and scalability exist, the RUP is an efficient integrated process for practical fields[8]. The concept of the various processes integration is similar to our process fragments integration, while the RUP is a pre-planned integration processes. The concept of our process complexity is based on more flexible model considering changes of development processes during a project execution. Our process complexity focuses on changes of an original development process by process fragments, and regarded as a development processes change.

Garcia et al. evaluated maintainability and modifiability of process models using new metrics based on GQM [9]. They focus on additional task as modifiability. The focus of additional task is similar to our concept of process complexity. Although their research target is theoretical process models, and our research target is practical development processes. In this way, there is few studies for measuring complexity of process during a project. Therefore, originality of our proposed complexity may be high.

3 Process Complexity Based on Process Fragment

3.1 Process Fragment

In software development, a manager makes a plan of a single development process like a waterfall process model at the beginning of a project, because developers and managers yet not have sufficient information about the development system at planning phase of a project. However, the planned single process usually continues to change until the end of the project. For example, at the beginning of a project, a manager makes a plan based on a waterfall process model. However the original process is changed to an incremental process model because several functions' development is shifted to next version's development. Moreover, at the requirement analysis phase, prototyping process may be added to an original process in order to satisfy customers' demands. If multiple releases like an incremental process model exist, developers have to implement new functions while developers correct faults that were caused in the previous version's development. In this paper, we call the original process "a base process", we call the additional process "a process fragment". While an original process is a process that was planned at the beginning of a project, a process fragment is a process that is added to the original process on the way of a project. "Fragment" means piecemeal process. Process fragments are simultaneously executed with a base process. Process fragment does not mean simple refinement of a base process, but rather may separately executes from a base process execution.

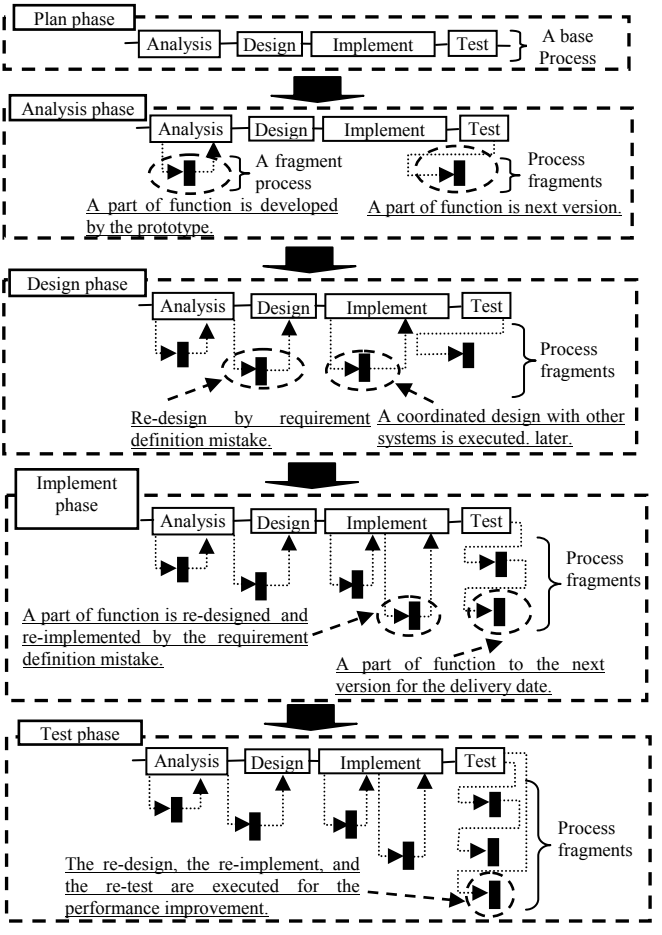

Fig. 1. A concept of process fragments

Figure1 shows an example of process fragment. At the planning phase, it is a simple development process that consists of analysis activity, design activity, implementation activity, and testing activity. In many cases, because of insufficient information, a manager often makes a rough simple process (macro process) rather than a detailed process (micro process) [10]. However, information about software increases as a project progresses, and the original simple process changes to more complicated processes. In the case of Figure 1, at the analysis phase, an unplanned prototype process was added to the original process because of customers' requests. As a result of analysis phase, implementations of some functions were shifted to next version development because of constraints of resources such as time, cost, and human. The process fragments were shown at the Figure1 as small black boxes. In the design phase, because customers detected miss-definitions of system specifications that were determined in the previous analysis phase, a process for reworking of requirement

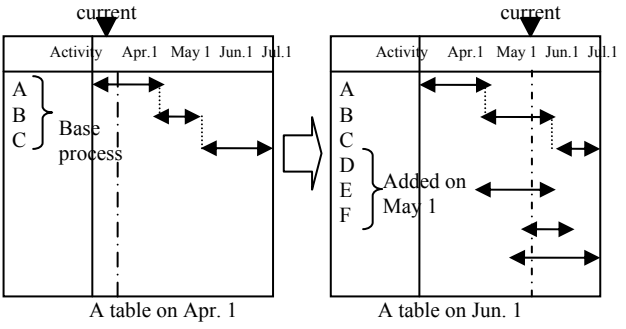

Fig. 2. Extracting process fragments from configuration of a workflow management table

analysis was added to the development process. Moreover, the manager shifted the development of a combination function with the outside system when the outside system was completed. During the implementation phase, several reworks of designs occurred. In the test phase, reworks of designs occurred because of low performance of several functions.

Process fragments are caused by urgent customers' requests, design errors, combination with outside systems on the way of development. Various sizes of process fragments exist. A small process fragment includes only an action such as document revision. A large process fragment may include more activities for example, a series of developing activities; design activity, implementation activity, and test activity.

3.2 Calculation of Process Complexity

3.2.1 Extracting Process Fragments

Process complexity is calculated based on process fragments. Process fragments are identified from a series of workflow management table. That is a continuator revised along the project. Figure 2 shows two versions of a workflow management table. Each column means a date, each row means an activity. A series of A, B, C activities is a base process. D, E, F activities are added to the base process on May 1. Therefore, D, E, F activities are process fragments. On Jun. 1, the base process and three process fragments are simultaneously executed. In this way, process fragments can be identified from configuration management of a workflow management table. Difference between current version and previous version of a workflow management table means process fragments. Of course, the proposed complexity is available in various development processes such as an agile process as long as managers manage process fragments in various management charts.

3.2.2 Definition of Process Complexity

Process complexity is defined by the two following equations.

$$PC_{(t)\_acummulate} = \sum_{i=1}^{N_{(t\_accumulate)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad (1)$$

$$PC_{(t)\_moment} = \sum_{i=1}^{N_{(t\_moment)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad (2)$$

$PC_{(t)\_accumulate}$: process complexity on time t including finished processes
$PC_{(t)\_moment}$: process complexity on just time t
$N_{(t\_accumulate)}$: the total number of process on time t including finished processes
$N_{(t\_moment)}$: the total number of process on just time t
$Num\_dev_{(t)i}$: the number of group of developers of the *i-th* process fragment on time t
$L_{(t)i}$: the number of simultaneous processes of the *i-th* process fragment on time t. But the *i-th* fragment is eliminated from these multiplications in $L_{(t)}$.
$term_{(t)i}$: ratio of an executing period of the *i-th* process fragment for the whole period of the project on time t, that is, if $term_{(t)i}$ is near 1, a executing period of the process fragment is near the whole period of the project.

Basically, we have two type process complexities. $PC_{(t)\_accumulate}$ is process complexity is accumulation of all process fragments including finished processes. $PC_{(t)\_moment}$ is a process complexity is on just time t not including finished processes. Finished process means that all tasks of the process already are completed. The reason of the two type complexities is based on different management viewpoints. If a manager wants to see the whole project characteristics, $PC_{(t)\_accumulate}$ is useful because the value of the complexity presents total accumulation of all process fragments. If a manager wants to see change of process on every day, $PC_{(t)\_moment}$ can be useful for grasping change of complexity on every day. For example, many process fragments occur at the first half of a project. Even if the process fragments have finished, the fragments' executions may harmfully influence products and process at the latter half of the project. In this management view, a manager uses a value of $PC_{(t)\_accumulate}$. On the other hand, tasks of every day change. The change of tasks of every day can be controlled by a value of $PC_{(t)\_moment}$

Theses process complexities basically depend on three elements; the number of group of the *i-th* process fragment on time t: $Num\_dev_{(t)i}$, the number of simultaneous processes of the *i-th* process fragment on time t: $L_{(t)i}$, and ratio of an executing period of the *i-th* process fragment for the whole period of project time t: $term_{(t)i}$. Granularity of group of developer($Num\_dev_{(t)i}$) depends on the scale of process fragments. If a process fragment is in detail of every hour, a group simply correspond to a person. If process fragment is large such as design phase, a group unit will be an organization such as SE group and programmer group. The granularity of group of developer will be carefully discussed in future research. The ratio of an executing period of the *i-th* process fragment for the whole period of project time t ($term_{(t)i}$) means impact scale of a process fragment. If a process fragment is very large, for example an executing period of the process fragment is almost same as the whole period of a project, the process fragment will influence largely the project. In contrast, if a process fragment is very small, for example an executing period is only one day, the process fragment will not influence a project so much.

In short, when more and larger scale process fragments are simultaneously executed, a value of process complexity becomes larger. When fewer and smaller scale process fragments simultaneously are executed, a value of process complexity becomes smaller. Values of the parameters of equation (1) and (2) are easily extracted from configuration management data of a workflow management table.

3.3 Setting a Base Process and Extracting Process Fragments

At the beginning of a project, a base process is determined. If a planned schedule is based on a typical waterfall process model such as the base process in Figure 1, the parameters' values of process complexity are $t=0$, $N_{(\square)}=1$, $Num\_dev_{(\square)}=3$ (SE group developer group, customer group), $L_{(\square)}=1$, *and* $term_{(\square)}=1$. Therefore the value of process complexity $PC_{(0)}=3$.

As a project progresses, unplanned process fragments are occasionally added to the base process at time *t1*. A manager registers the process fragments as new activities to the workflow management table. The manager also assigns developers to the new activities. Here, we assume that the planned period of a base process is 180 days. A manager adds two activities to the workflow management table. The period of each additional activity is planned as 10 days. Therefore the total number of process $N_{(t1)} = 3$, and the process complexity is calculated as follows;

(1) for $i = 1$ (a base process)

- $Num\_dev_{(t1)} = 3$
- $L_{(t1)1} = 3$
- $term_{(t1)1} = 180/180 = 1.0$

(2) for $i = 2$ (the first process fragment)

- $Num\_dev_{(t1)2} = 1$
- $L_{(t1)2} = 3$
- $term_{(t1)2} = 10/180 = 0.056$

(3) for $i = 3$ (the second process fragment)

- $Num\_dev_{(t1)3} = 1$
- $L_{(t1)3} = 3$
- $term_{(t1)3} = 10/180 = 0.056$

Finally, a value of process complexity at $t=t1$ can be calculated as $PC_{(t1)\_moment} = 9.000 + 0.168 + 0.168 = 9.336$. In this way, the value of process complexity at time t can be calculated based on workflow management table.

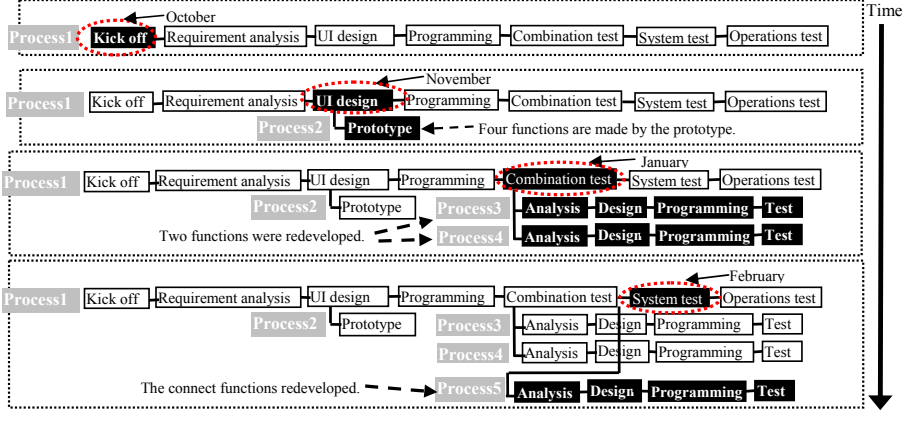

Fig. 3. A variation of development process of the HInT V2 project

4 Application to Six Industrial Projects

The process complexity has been applied to six practical projects; two versions of HInT project [13] and four versions of p-HInT project [11][12]. These projects executed by a same organization of a system development company. One of the 6 projects is presented at the following subsection.

4.1 The HInT V2 project

The version 2 of HInT project developed a web-based educational portal system. The development began from October 2007, the release of the HInT was April 2008. Because the workflow management table was updated every week, 20 versions of the workflow management table are obtained. At the beginning of the project, the number of activities in the workflow table was 20. At the end of the project, the number of activities reached to 123. Each activity had a planned schedule and a practice result of

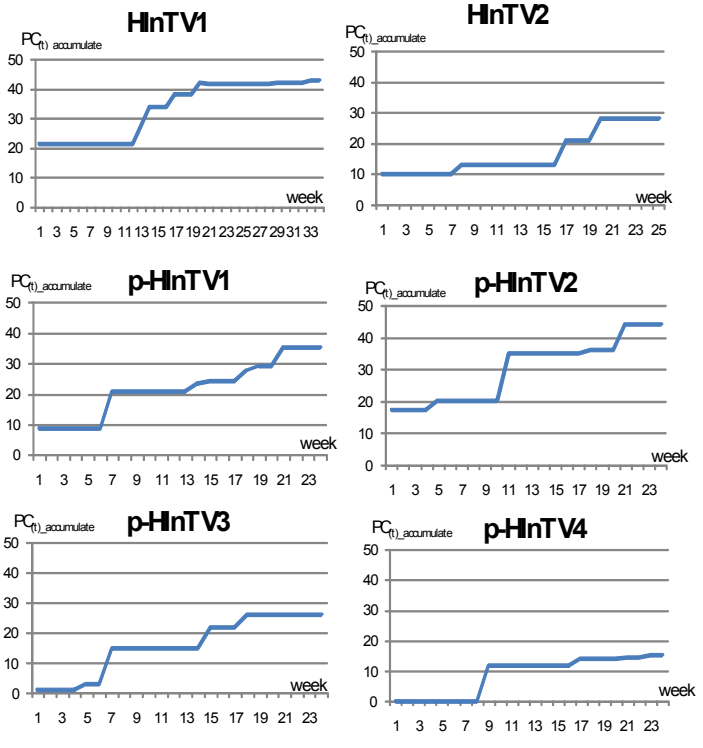

Fig. 4. Changes of values of process complexities (accumulate version) of 6 projects

execution. Figure 3 shows a rough variation of development process of the project. At the beginning of the project, the shape of the development process was completely a waterfall process. However, at the UI design phase, a prototype process was added to the waterfall process. In the prototype process, four trial versions were presented to customers. At the combination test phase, developers and customers found significant

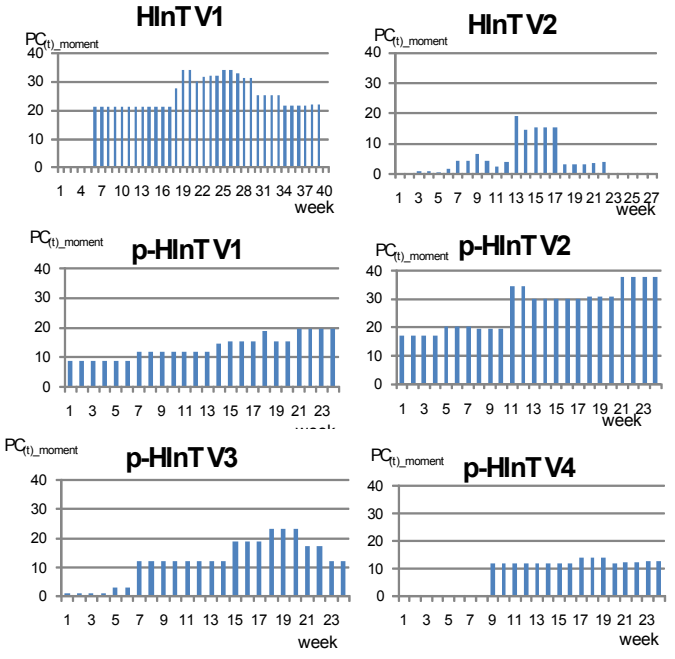

Fig. 5. Changes of values of process complexities (moment version) of 6 projects

errors of specifications and two process fragments were added to the development process in haste. Reworks such as re-design, re-implement, and re-test for the specification errors continued until the operation test phase. At the system test phase, an error of a function connecting with an outside system occurred and a new process fragment was introduced to the development process. The introduced process fragment consists of various activities such as investigating network environments, investigating specification of the outside system, and revising the programs. These activities continued until the delivery timing.

4.2 Changes of Process Complexities of the 6 Projects

Figure 4 shows the changes of process complexities (accumulate version) of the 6 projects. Figure 5 shows the changes of process complexities (moment version) of the 6 projects. The highest value of process complexity (accumulate version) is *44.4* of the p-HInT V2 project. A minimum value of process complexity (accumulate version) is *15.4* of the p-HInT V4 project. Therefore, the p-HInT V2 project included a complicated base process and many process fragments. The p-HInT V4 project consisted of a simple base process and few process fragments.

On the other hand, the changes of process complexities (moment version) in Figure 5 present different trends. Process complexities (moment version) of HInT V1 and p-HInT V2 were relatively high during the projects. That is, many fragment processes concurrently executed in HInT V1 and p-HInT V2. HInT V1 project was a new

development system. Because customers could not image the new system, customers' requests frequently changed. Therefore, developers should concurrently execute various development activities including activities for customers' new requests. In contrast, process complexities of HInT V2 keep low during the project. Customers could easily image new functions because customers mastered the original system. In p-HInT project, process complexities of p-HInT V2 were high. In the p-HInT V2, many system troubles occurred. The troubles were caused by development of p-HInT V1. Therefore, developers should concurrently execute not only development of new functions of version2 but also debugging activities of the errors. The p-HInT V2 project had complicated processes with many process fragments. After that, in p-HInT V3, the manager decided a product refactoring without developing new functions. Process complexities were relatively low in the p-HInT V3. Because the product refactoring executed smoothly in the p-HInT V3, process complexities of p-HInT V3 and p-HInT V4 did not become so high during the projects.

4.3 A Trial Tool for Visualizing Process Complexity

We have developed a trial tool for visualizing changes of process complexity (moment version) during a project. Figure 6 shows images of the changes in three projects; HInT V1, p-HInT V2, p-HInT V3. One block means one process fragment. A size of block means complexity of a process fragment. A big block is more complicate than a small block. In addition, X axis means *time*, Y axis means accumulated values of $L_{(t)}$ of equation (2) at time t, Z axis means accumulated values of $Num\_dev_{(t)}$ of equation (2) at time t. That is, if a block is long X-axially, the process fragment has long development time. If a block is long Y-axially, the process fragment has to execute concurrently with many other processes. If a block is long Z-axially, the process fragment has to execute with many development groups. In HInT V1, many fragment processes concurrently executed on the latter half of the project. Therefore, many blocks were piled, moreover, the blocks were long Y-axially. However because the blocks of HInT V1 is short X-axially, the process fragments finished quickly. In addition, concurrent executed process fragments of HInT V1 are more than one of p-HInT V2 and p-HInT V3. The piled process fragments of HInT V1 are higher than the piled process fragments of p-HInT V2 and p-HInT V3. In this way, characteristics of process complexity of each project are visualized in the tool.

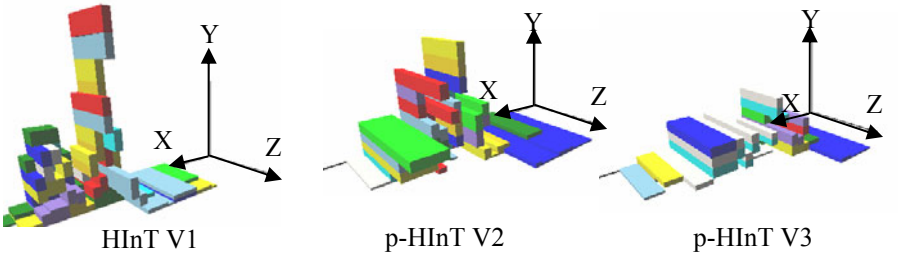

Fig. 6. Visualized images of changing process complexities

5 Process Complexity – Product Quality (PCPQ) Model

We propose a Process Complexity – Product Quality (PCPQ) model. A PCPQ model is built by pairs of a final process complexity and specific gravity of failure in several projects. A final process complexity means a value of process complexity (accumulate version) at the ending of project. A specific gravity of failure means a product quality based on importance and the number of post-release failures. A PCPQ model is built on each organization because a way of creating workflow management tables and management of post-release failure is different. For example, a manager of an organization creates a daily workflow management table. In this case, processes are divided into small process fragments. Scale of process depends on a management way of each organization. In addition, a way of management of failures is different among organizations. For example, a manager decides that a system-down error is a most important failure. Another manager decides that requirement analysis error is a most important failure. The way of deciding importance of failures is different on each organization. Therefore, a PCPQ model is built by each organization.

After building a PCPQ model, a manager can predict a post-release product quality even if the project is not finished. On the way of a project, a value of process complexity (accumulate version) can be calculated based on a scheduled workflow management table. When a manage remake a plan for process fragments, a post-release product quality can be predicted based on the PCPQ model.

5.1 A Procedure of Building a PCPQ Model

A PCPQ model is built in the following procedure;

Step1: Preparing several finished projects in a same organization.
Step2: Calculating a value of final process complexity (accumulate version) of each project from workflow management tables.
Step3: Collecting post-release failures, deciding importance of the failures.
Step4: Calculating a value of specific gravity of failure by each project.
Step5: Making an approximate value curve of the relation between the process complexities and the specific gravity of failures.

The approximate value curve of the relation between the process complexities and the specific gravity of failures means a PCPQ model. The next subsection shows an example of making a PCPQ model based on the 6 projects of section 4.

5.2 An Example of Making a PCPQ Model

Step1:We prepare six finished projects; HInT V1, HInT V2, p-HInT V1, p-HInT V2, p-HInT V3, p-HInT V4 in a same organization.

Step2:Six final process complexities (accumulate version) are calculated. Table 1 shows the values of the final process complexities.

Step3:We collected post-release failures. Each failure's importance was decided. The Table 1 also shows the number of the failures with importance ranks; SS, S, A, B, C.

In the organization, the failures are categorized into 5 ranks. SS rank means system-down level error. S rank means errors of missing important functions and performance. A rank means errors of missing middle important functions, B rank means low quality of user-interface, C rank means errors of presentation such as messages and button names. The failures were accumulated on failure management tables on each project after release. The importance rank of each failure is decided by the manager and customers.

Step4:We calculated values of specific gravity of failures. The specific gravity of the failures is a value that multiplied the number of failures and the importance rank. In the calculation, we set up that a constant of SS is 5, a constant of S is 4, a constant of A is 3, a constant of B is 2, and a constant of C is 1. For example, a value of specific gravity of HInT V1 is calculated by "5*4 + 4*4 + 3*6 +2*9 + 1*11". The value is 83. In the same way, the specific gravity of failure of HInT V2 is 66, one of p-HInT V1 is 82, and one of p-HInT V2 is 82, one of p-HInT V3 is 88, one of p-HInT V4 is37, and one of p-HInT V5 is 28.

Step5:We plotted relations between process complexities and specific gravity of failures. Figure 7 shows the relations and an approximate value curve. The approximate value curve is as a follow;

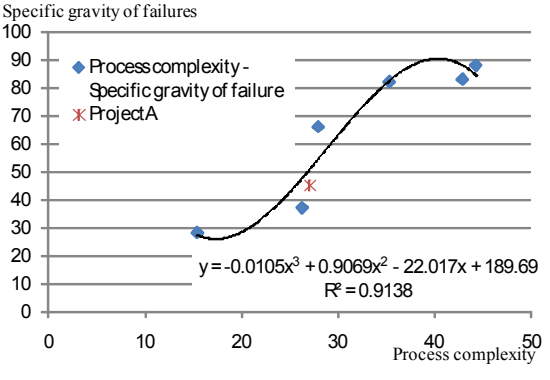

Fig. 7. An approximate value curve of relation between process complexities and specific gravities of failures

$$y = -0.0105X^3 + 0.9069X^2 - 22.017X + 189.69 \quad (3)$$

y: a value of specific gravity of failures
x: a value of final process complexity (accumulate version)

Equation (3) is a PCPQ model of the organization. Coeffient of determination (R-squared) is 0.9138. Of course, if a value of coeffient of determination is too small,

the PCPQ model is meaningless. Although the PCPQ model is based on a polynomial expression, the relation can present using linear approximation, logarithm approximation, and exponential approximation.

5.3 Predicting Product Quality Based on the PCPQ Model

Using the PCPQ model of Figure 7, product quality of another project is predicted. A target project is "Project A" developing an e-learning system in the same organization. Of course Project A is different from the six projects. Change of process complexity is shown Figure 8. At the thirteenth week, the value of process complexity is *25.21*. The value of the process complexity is applied to the PCPQ model (Equation(3)). Therefore, a value of specific gravity of failure is calculated as *42.78*. Therefore, the manager can predict post-release product quality as not better than p-HInT V3, and better than HInT V2.

The prediction of Project A is evaluated using real failure data. The number of post-release failure of Project A is *20*. The number of failure with SS and S rank is 0, the number of failure with A rank is 8, the number of failure with B rank is 9, and the number of failure with C rank is 3. The value of specific gravity of failure is *45*. The prediction value based on the PCPQ model is *42.78*, the real value is *45*. A plot "*" in Figure 7 is a relation between the process complexity and the specific gravity of failure of the Project A. We may judge that the product quality prediction of Project A at the thirteenth week is useful.

Table 1. Parameter values of a PCPQ model

Project	Final process complexity	Specific gravity of failures	Importance of failure				
			SS	S	A	B	C
HInT V1	43.0	83	4	4	6	9	11
HInT V2	28.0	66	2	1	6	10	13
p-HInT V1	35.4	82	2	3	19	1	1
p-HInT V2	44.4	88	10	6	4	1	0
p-HInT V3	26.3	37	3	3	2	1	2
p-HInT V4	15.4	28	0	0	7	3	1

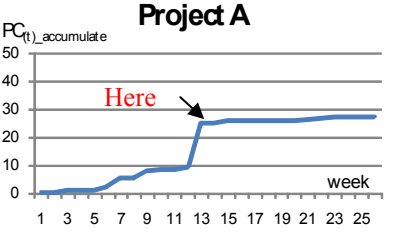

Fig. 8. Change of process complexity (accumulate version) of Project A

6 Summary

We propose a metric of process complexity based on process fragments. the process complexity has three elements; the number of group of developers, the number of simultaneous process, and ratio of an executing period for a period of the whole project. Process complexity can present changes of development processes with additional and piecemeal process fragment during a project. Process complexity is applied to six industrial projects. We could grasp how the development processes of the six projects became complicated as the projects progressed. In addition, we also proposed a way of making a PCPQ (Product Complexity-Product Quality) model. A PCPQ model is useful to predict post-release product quality on the way of a project. The PCPQ model is an approximate curve of a relation between process complexity and product quality. The model is built using process complexity and actual product quality of finished projects. A PCPQ model using the six projects' data was built. As a result, a post-release product quality was able to be predicted by the PCPQ model.

In future, we will evaluate several PCPQ models. A PCPQ models is built every organization because management ways of workflow management tables and post-release product quality are different among organizations. Therefore, we need building PCPQ models in different organizations. Then usefulness of PCPQ models will be evaluated.

References

1. Royce, W.: Software Project Management: A unified Framewaork. Addison-Wesley Professional, USA (1998)
2. Guide to the Software Engineering Body of Knowledge (SWEBOK), http://www.computer.org/portal/web/swebok
3. Humphrey Watts, S.: Managing the software process. Addison-Wesley Professional, USA (1989)
4. Sakamoto, K., Tanaka, T., Kusumoto, S., Matsumoto, K., Kikuno, T.: An Improvement of Software Process Based on Benefit Estimation (in japanese). IEICE Trans. Inf. & Syst. J83-D-I(7), 740–748 (2000)
5. Cugola, G.: Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models. IEEE Transaction of Software Engineering 24(11), 982–1001 (1998)
6. Fuggetta, A., Ghezzi, C.: State of the art and open issues in process-centered software engineering environments. Journal of Systems and Software 26(1), 53–60 (1994)
7. Kruchten, R.: The Rational Unified Process. Addison-Wesley Professional, USA (2000)
8. Manzoni, L.V., Price, R.T.: Identifying extensions required by RUP (rational unified process) to comply with CMM (capability maturity model) levels 2 and 3. IEEE Transactions on Software Engineering 29(2), 181–192 (2003)
9. Garcia, F., Ruiz, F., Piattini, M.: Definition and empirical validation of metrics for software process models. In: Proceedings of the 5th International Conference Product Focused Software Process Improvement, pp. 146–158 (2004)
10. Obana, M., Hanakawa, N., Yoshida, N., Iida, H.: Process Fragment Based Process Complexity with Workflow Management Tables. In: International Workshop on Empirical Software Engineering in Practice, pp. 7–12 (2010)

11. Hanakawa, N., Yamamoto, G., Tashiro, K., Tagami, H., Hamada, S.: p-HInT: Interactive Educational environment for improving large-scale lecture with mobile game terminals. In: Proceedings of the16th International Conference on Computers in Education, pp. 629–634 (2008)
12. Hanakawa, N., Obana, M.: Mobile game terminal based interactive education environment for large-scale lectures. In: Proceeding of the Eighth IASTED International Conference on Web-based Education, pp. 7–12 (2010)
13. Hanakawa, N., Akazawa, Y., Mori, A., Maeda, T., Inoue, T., Tsutsui, S.: A Web-based integrated education system for a seamless environment among teachers, students, and administrators. International Journal of System & Computer in Japan 37(5), 14–24 (2006)