A Model of Project Supervision for Process Correction and Improvement

Masateru Tsunoda, Akito Monden, Tomoko Matsumura[1], and Ken-ichi Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
{masate-t, akito-m, tomoko-m, matumoto}@is.naist.jp

Abstract—**Recently, software functional size becomes larger, and consequently, not only a software developer but also a software purchaser suffers considerable losses by software project failure. So avoiding project failure is also important for purchasers. Project supervision (monitoring and control) is expected for the purchaser to suppress risk of project failure. It is performed by sharing software metrics during the project for the purchaser to grasp the status of the project, and corrective actions are done based on analysis results of the metrics. Although there are some software measurement models, the models are not enough to describe how to confirm effects of project supervision. To acquire the effects certainly, the purchaser and the developer should quantitatively confirm whether the effects are acquired or not by project supervision. In addition, the models cannot represent corrective actions when symptoms of project failure are found. We propose the model for project supervision. The model explains planning, collecting data, transforming data, analyzing data, reaction toward found issues, and confirming effect of project supervision. With our model, project supervision can be described more rigorously.**

Keywords-risk management; measurement; Fault Tree Analysis; Business Process Modeling Notation; corrective action

I. INTRODUCTION

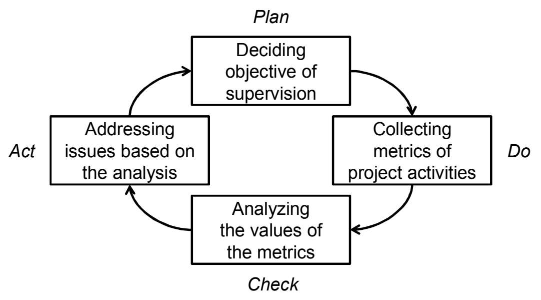
Recently, software functional size becomes larger, since software is used in various situations and needed for more functions. Consequently, not only a software developer but also a software purchaser suffers considerable losses if software project is failed (delay of delivery time, project cost overrun, or insufficient quality of developed software are occurred). So avoiding project failure is more important for purchasers than before. Project analyzing with a purchaser and a developer is expected to be effective way for the purchaser to suppress risk of project failure [14], especially when developer's project management skill is insufficient. Project analyzing with a purchaser and a developer is performed by sharing software metrics data, and based on the results, addressing issues such as too complex source code or insufficient unit test is conducted. In this paper, we call the activities *project supervision*. The concept of project supervision is monitoring and control.

Besides, in recent years, software developers outsource a part of software development to subcontractors (e.g. offshore developers in India or China) because of lack of human resources, or pressure of restraining software development cost. However, subcontractors' project failure sometimes causes primary contractor's project failure. Project supervision with a primary constructor (purchaser) and a subcontractor (developer) is also expected to be effective.

Appropriate model of project supervision is expected to bring benefits to the purchaser and the developer. The model defines elements of the activities and their relationships. More intuitively, the model clarifies how to perform project supervision. By the model, it becomes easy to make a rigorous plan of project supervision with less effort. The plan enforces steady implementation of project supervision, and it drives success of the project. In addition, the model can be used as a template to make a catalog of project supervision. The catalog enables reuse of know-how of project supervision, and a company which has little experience of data analysis easily introduces project supervision. The role of the catalog is similar to design pattern [10].

The model is required to describe project supervision activities. Applying plan-do-check-act (PDCA) cycle, project supervision has four main activities, i.e. deciding objective of supervision (plan), collecting metrics of project activities while they are performed (do), analyzing values of metrics (check), and addressing issues based on the analysis (act), as shown in Fig. 1. For instance, "Evaluate source code quality" is set as the objective of supervision, and source code complexity metrics (e.g. cyclomatic complexity) are chosen on plan phase. Then, (source code is made and) the metrics are measured on do phase, and their values are compared with certain criteria on check phase. On act phase, modules are modified if complexity metrics do not meet the criteria, because it suggests source code quality is not high.

Although there are some models for software


Plan
Deciding objective of supervision
Do
Collecting metrics of project activities
Check
Analyzing the values of the metrics
Act
Addressing issues based on the analysis
Figure 1. PDCA cycle of project supervision.

[1] Presently with Hitachi Ltd.

measurement [6][15][20][22][24], the models are not enough to describe how to confirm effects of project supervision. In most cases, objective of supervision is set to aim some effects. For example, an objective of supervision "Evaluate source code quality" is set to make modifications of source code easy, and that is expected to prevent consuming time to modify source code and schedule delay. To acquire the effects certainly, the purchaser and the developer should quantitatively confirm whether the effects are acquired or not after addressing issues (the act phase). The confirmation is performed using certain metrics in later process or project, and it promotes rigorous project supervision. Capability Maturity Model Integration (CMMI) mentions necessity of the confirmation in Project Monitoring and Control (PMC) process area [4]. So, the model of project supervision is required to describe how to confirm the effects.

In addition, the existing models for software measurement cannot represent activities of act phase. For example, Chirinos et al. [6] proposed the software measurement model which defines elements such as type of metrics, range of value, and measurement method, but it does not include elements related to act phase. To get the effects of project analyzing and addressing issues, activities of act phase is indispensable, and therefore, description capability of the act phase is required for the model of project supervision. CMMI indicates necessity of corrective actions in PMC process area [4]. Also, the description is found in other areas of software engineering. For example, the Hazard Analysis and Critical Control Point (HACCP; management system for food safety) [9], good manufacturing practice (GMP; guidance for manufacturing pharmaceutical products) [28], and ISO 9001 [16] require corrective action.

In this paper, we propose a new model to describe activities of project supervision precisely. The model illustrates activities of correcting data, analyzing data, addressing issues, and evaluating effects. Relationships between the issues are shown by fault tree analysis (FTA) [8] based figure. In addition, to promote rigorous project supervision, the model denotes procedure of the activities using Business Process Modeling Notation (BPMN) [25].

The major contribution of our paper is introducing elements about addressing issues and evaluating effects to project supervision model, and FTA is used to illustrate relationships among analyzing data, addressing issues, and evaluating effects. Existing models cannot signify the relationships explicitly. With our model, a rigorous plan of project supervision can be made with less effort. It promotes steady implementation of project supervision, and that brings project success. The efficiency of the model is similar to the entity-relationship model for database [5] or UML (Unified Modeling Language) in software design.

In what follows, Section II clarifies requirements for modeling project supervision. Section III explains structure of the model. Section IV introduces comparison of other models. In the end, Section V concludes the paper with a summary.

II. REQUIMENTS FOR MODELING PROJECT SUPERVISION

We identify requirements for the model of project supervision by seeing activities of it. Project supervision is classified into two types. One type is applied to ongoing project to control it. It is used to grasp project status, and to perform corrective action during the project. The aim is to avoid project failure such as schedule delay or low quality software. We call the type *in-process supervision*.

The other type of project supervision is applied to finished project. Preventive action (it prevents emergence of issues in the next project) is conducted by analysis results of the project. The aim is to avoid next projects' failure. We call the type *post-process supervision*. Post-process supervision treats data which is difficult to measure before finishing the project (e.g. number of failures after software release), or treats improvement activity which is difficult to apply during project (e.g. drastic change of development process).

Standard procedure of in-process supervision is as follows:

1. Plan of in-process supervision is made. Objective of supervision is set under an agreement between a developer and a purchaser.
2. Values of measures are collected by hand or tool from ongoing project.
3. Indexes or figures are made from the collected measures.
4. Check whether problematic events occur or not based on the indexes or the figures.
5. Corrective actions toward the problematic events are performed.
6. Project status is confirmed in later process or after the project finished.

Standard procedure of post-process supervision is almost same as in-process procedure. It is as follows:

1. Plan of post-process supervision is made. Objective of supervision is set under an agreement between a developer and a purchaser.
2. Values of measures are collected by hand or tool from finished project.
3. Indexes or figures are made from the collected measures.
4. Identify problems to be improved based on the indexes or the figures.
5. Preventive actions are conducted to suppress the problems.
6. Next project status is confirmed by some indexes or in-process supervision.

To describe procedure of project supervision, a model is required to present:

Req. 1 Objective of project supervision.
Req. 2 How to collect measures.
Req. 3 How to make indexes and figures from the collected measures.
Req. 4 How to analyze the indexes and the figures.
Req. 5 Which corrective actions or preventive actions should be performed based on the analysis.
Req. 6 How to confirm effects of the project supervision.

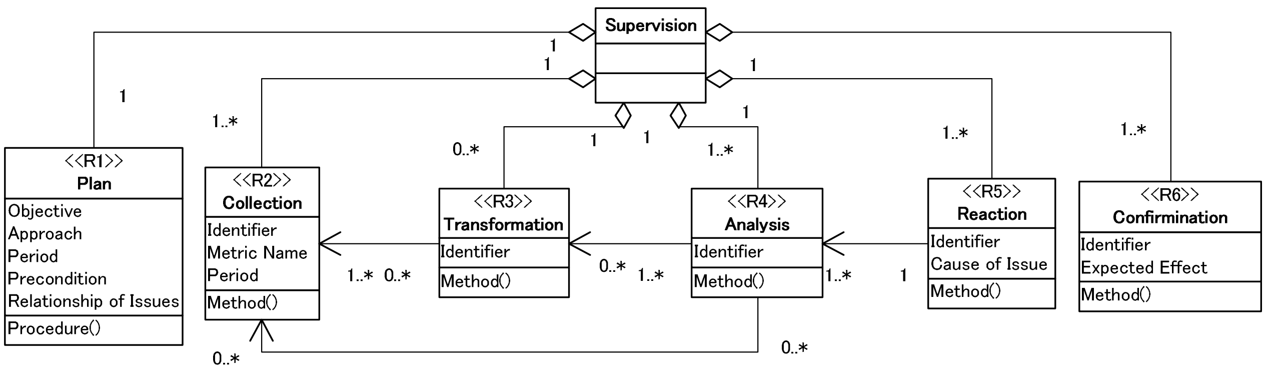


Figure 2. Structure of the model of project supervision.

Each requirement corresponds to each step of the procedures.

III. Structure of Project Supervision Model

To satisfy the requirements stated in section II, we propose a model for project supervision. In the model, project supervision consists of *Plan*, *Collection*, *Transformation*, *Analysis*, *Reaction*, and *Confirmation*. The elements correspond to the requirements, and they explain project supervision activities. Fig. 2 illustrates the structure of the model using a class diagram of UML (Unified Modeling Language). The elements and the requirements are expressed by classes and stereotypes respectively. The descriptions about implementations are signified by methods of the classes, and other descriptions are signified by attributes. Typically, in a measurement plan based on our model, three to five metrics are set for one objective.

Details of elements and relationships of them are explained below. Note that there are examples of the elements in the explanations, but they are not connected to each other (That is, they have different objective of supervision), for we prioritized to select most understandable examples to explain the elements.

A. Plan

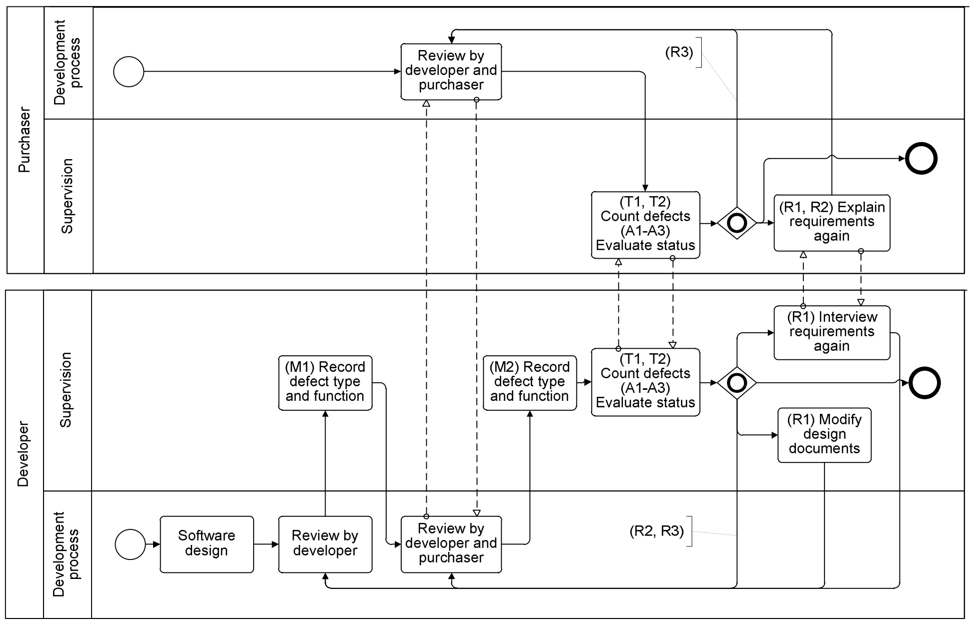
The plan includes information which is needed when making a plan of the project supervision. This element has five items explained below.

- *Objective* is purpose of the project supervision, and it denotes what is clarified by the supervision.
- *Approach* is brief explanation of how to analyze data.
- *Period* indicates when transforming data, analyzing data, and addressing issues are performed. The activities are done almost at the same time.
- *Precondition* is requirements which a project or an organization should be fulfilled before the supervision is applied.
- *Procedure* illustrates the supervision workflow (i.e. when activities such as analyzing or corrective action are performed) using Business Process Modeling Notation (BPMN) [25]. In BPMN, *Pool* shows an organization, and *Lane* does a role or a department. We used the pool to signify a developer and a purchaser, and the lane is used to distinguish development process and supervision process. The identifiers of other elements are shown in the procedure (The identifier is explained in the next element).
- *Relationship of issues* illustrates relations of issues addressed by the supervision. They are expressed using the notation of Fault Tree Analysis (FTA) [8]. This item contains the identifiers of other elements. Although the confirmation indicates positive effects which are gained by removing issues found by project supervision, this item shows negative effects which occur if the found issues are not addressed. This is because FTA is used to express relationships of causes of faults.

The approach and the precondition are used when making a catalog of project supervision. They are useful information for a developer and a purchaser to select a case included in the catalog. The approach helps to grasp overview of the case at a glance, and the precondition is used whether the case can be applied or not. On the contrary, they are not used to make a plan of project supervision, because there is no need to select a case included in the catalog at the time.

There is only one plan for an instance of project supervision, because project supervision activities are decided by the objective included on the plan. An example of the procedure is shown in Fig. 3. BPMN signifies start of process by a circle, and end of process by a circle with a bold border. A square with rounded corners indicates a task, and a diamond does a branch. Large box is the pool, and nested box is the lane. Parenthesized characters are the identifier of other elements.

Fig. 4 is an example of the relationship of issues. In FTA, a box means an issue, and a circle indicates cause of issues. Upper events are occurred by lower events, and the relationship of them is shown using Boolean logic such as OR gate. Although our model uses the diagram of FTA, it is not necessary to perform FTA. We adopt the notation because it fits our model very well. Besides FTA, root cause analysis (RCA) may be useful to make the figure. Note that it is not required that all relationships between causes and results are illustrated by the figure. The figure is used to



Purchaser
Development process
Supervision
Developer
Review by developer and purchaser
(R3)
(T1, T2) Count defects (A1-A3) Evaluate status
(R1, R2) Explain requirements again
(R1) Interview requirements again
(M1) Record defect type and function
(M2) Record defect type and function
(R1) Modify design documents
Software design
Review by developer
(R2, R3)

Figure 3. An example of the procedure included on the plan.

denote why the metric is used. That is, it shows what kind of issue is clarified by analyzing the metric, what to do when an issue is found, and what is expected to control the metric.

An example of descriptions included in the plan is shown below (Note that the objectives of Fig. 3 and Fig. 4 are different from the example).

Objective: evaluate project progress
Approach: use transition data of program size (LOC) and number of found defects
Period: regularly repeated (e.g. performed at project status meeting) during coding and testing
Precondition: bug tracking system (BTS) and software configuration management (SCM) are introduced, and bug report and source code are registered to the system promptly (not registered them in bulk on weekend, for example)

B. Collection

The collection explains measurement method of measures used on the project supervision (The word "measure" indicates base measure or derived measure (metric) [15]). Four items are included in the element. Details of them are explained below.

- *Identifier* is used to be referenced from other elements. A combination of characters and numbers such as "M1" is used as a value of the identifier.
- *Measure name* is name of collected measure such as cyclomatic complexity. The measure name is not indispensable to be referenced by other elements, because the collection has the identifier. We include this item to the model, since some common measure names are expected to help to understand definition of the measure.
- *Period* indicates when the measure is collected. While timing of transforming data, analyzing data, and addressing issues is shown on the plan, it does not include timing of collecting data. This is because timing of collecting data is different for measures, and therefore it should be shown by each measure using this item.
- *Method* denotes how to collect measures. In this item, *measurement target* indicates a target which is measured to get a value of a measure, and shows extraction condition. Also in the item, tool is indicated if it is needed to get the values.

One or more of the collection is included in an instance of supervision. The collection is expressed by tabular form. Table I is an example of the collection (In the example, the number of the collection is three).

C. Transformation

The transformation shows how to transform collected data into indexes or figures to analysis it. Transforming data is performed just before analyzing data. In the element, there are two items explained below.

- *Identifier* has the same role as the identifier of the collection.
- *Method* explains how to transform measures defined in the collection into indexes or figures, using a text and an example of the figure.

An instance of supervision includes zero or more of the transformation. If transforming data is not necessary (That is, measures defined on the collection is directly used in the analysis), the number of the transformation is zero.

We put a directed association between the collection and the transformation as shown in Fig. 2, since the transformation refers to measures defined in the collection.

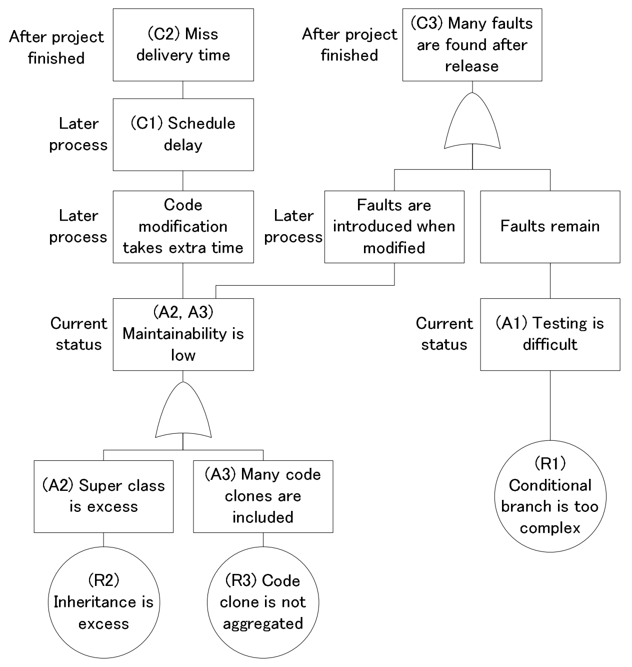

After project finished
(C2) Miss delivery time
After project finished
(C3) Many faults are found after release
Later process
(C1) Schedule delay
Later process
Code modification takes extra time
Later process
Faults are introduced when modified
Faults remain
Current status
(A2, A3) Maintainability is low
Current status
(A1) Testing is difficult
(A2) Super class is excess
(A3) Many code clones are included
(R1) Conditional branch is too complex
(R2) Inheritance is excess
(R3) Code clone is not aggregated

Figure 4. An example of the relationships of issues.

We set the multiplicity of the collection, considering that a measure on the collection is used in some indexes or figures on the transformation, or a measure is not used by it (directly used in the analysis). Also, the multiplicity of the transformation is set, taking into account that an index or a figure is made from some measures.

An example of the figure in the method is illustrated in Fig. 5, and the text is shown below. In the examples, used measures are same as the example of the collection shown in Table I. Parenthesized characters at the beginning of the example signify the identifier of the transformation.

(T1) **Method:** To make a line chart of the cumulative number of faults (M1) and the number of unresolved faults (M2), horizontal axis is set as data collected date, and vertical axis is set as the number of faults. To make a bar chart of average retention time (M3), horizontal axis is set as data collected date, and vertical axis is set as average retention time. And then the chars are overlapped.

D. Analysis

The analysis clarifies how to check current status (i.e. whether issues occur or not in the project) using indexes and figures explained on the transformation. This element has three items explained below.

- *Identifier* has the same role as the identifier of the collection.
- *Method* describes how to analyze indexes and figures explained on the transformation. The analysis is done by checking trend of the figure against particular patterns, or by comparing the index to a certain reference value. Also, the method describes what the analysis result suggests.

Basically, one analysis is described for one issue, because one reaction corresponds to the issue in the model, and it makes linking the analysis, the issue, and the reaction easy. If analyzed target (the measure, the index or the figure) is different but the identified issue (analysis result) is same, each analysis is described separately, because it makes description of analysis simple. For example, schedule delay (issue) is recognized by two different figures, each analysis is described independently. Also, if one analysis result suggests two or more issues, both are described in the analysis. For instance, if one analysis result suggests that purchaser's explanation of requirements is insufficient, or software design documents made by the developer are incomprehensible for the purchaser, but it is difficult to identify which is occurred by the result, both are written in the analysis.

One or more issues are recognized by analyzing one index or one figure explained on the transformation, or one measure defined on the collection. So, we put a directed association between the analysis and the collection, and between the analysis and the transformation as depicted in Fig. 2. Also, as indicated in Fig. 2, the multiplicity of the analysis is set.

Usually, to recognize the issue, one index, one figure, or one measure is used, and sometimes two or more of them are used at the same time. Additionally, the analysis does not use the index and the figure, and use the measure only in some cases. On the contrary, the analysis does not use the measure, and use the index or the figure occasionally. So, we set the multiplicity of the collection and the transformation as shown in Fig. 2.

The following is an example of the description of the analysis. In the example, code clone [18] means duplicate code, and it harms software maintainability, because two or

TABLE I. AN EXAMPLE OF THE COLLECTION.

ID	Measure name	Period	Method
M1	Cumulative number of faults	During testing	• **Measurement target:** faults recorded in bug tracking system (BTS) • Sum number of faults by software function regularly. ○ In addition, sum number of faults by severity if severity is recorded.
M2	Number of unresolved faults	During testing	• **Measurement target:** faults recorded in BTS, and their status is not close. • Sum number of faults by software function regularly. ○ In addition, sum number of faults by priority if priority is recorded.
M3	Average retention time	During testing	• **Measurement target:** faults recorded in BTS, and their status is not closed. • Compute average of elapsed time from date reported by software function regularly. ○ In addition, eliminate faults whose priority is low before computing the average, if priority is recorded. This is because low priority faults are not modified quickly, and it makes average retention time long.

more duplicate code should be modified. Parenthesized characters at the beginning of the description signify the identifier.

(A3) Method: Modules are regarded as having too many code clones, when their code clone content rate (M3) are more than 20%, and they are neither GUI modules nor automatically generated modules. Maintainability of the modules is considered to be low.

E. Reaction

The reaction suggests causes of the issues found in analyzing, and indicates how to address the issues to achieve the objective in the plan. Three items are included in the element. Details of them are explained below.

- *Identifier* has the same role as the identifier of the collection.
- *Cause of issue* denotes suggested causes of found issues. In Fig. 6, "Code clone is not aggregated" is an example of the cause of issue. When additional analysis is needed to identify the cause, the analysis is describes as other project supervision. For example, when analyzing change history of requirement is needed to identify causes of schedule delay, supervisions of them are made independently, and this item mentions that causes of schedule delay is recognized using supervision of change history of requirement.
- *Method* explains corrective actions for in-process supervision and preventive actions for post-process supervision. In the item, *Actor* denotes who performs the method. Developer, purchaser, or developer and purchaser is set as the actor. Typically, the method describes how to eliminate causes, and for some a certain kind of issue, it does how to mitigate harmful effects of the issue. For instance, in Fig. 6, if "Code modification takes extra time" is harmful effect of the issue, "Aggregate code clone" is eliminating the cause, and "Use the clone detection tool" is mitigating the effect (The clone detection tool [23] finds duplicated code quickly, and it reduces effort of code modification and probability of overlooking duplicated code to be modified).

One reaction responds to one cause found by analyzing. In some cases, one reaction relates to two analyses, because when analyzed target is different but the identified issue is same, each analysis is described separately, as mentioned previously. So, we set a directed association between the analysis and the reaction, and their multiplicities, as indicated in Fig. 2. If analysis result is not a twofold situation (good or bad), but a sort of 'traffic lights,' several possible actions are written with the conditions. he following is an example of the reaction. The identifier of the reaction is parenthesized characters at the beginning of the example.

(R3) Cause of issue (A3): There may be many code clones which can be aggregated.
Method
- **Actor:** developer
- Check code clones to examine whether they can be aggregated or not, and aggregation is done if it does not raise harmful effect. If it is difficult to aggregate code clones, the clone detection tool is used when source code is modified on later process.

F. Confirmation

The confirmation explains expected effects by project supervision, and how to confirm them. In the element, there are four items explained below.

- *Identifier* has the same role as the identifier of the collection.
- *Period* denotes when the expected effect is observed. For in-process supervision, "in later process" or "after project finished" is set, and for post-process supervision, "in next project" or "after next project finished" is set. The reason why this item is needed is almost same as the period in the collection.
- *Expected effect* describes effects which are expected to be gained from project supervision. For instance, as indicated in Fig. 7, when the objective is "Evaluate project progress," the expected effects are "Suppress schedule delay" and "Suppress to miss delivery time."
- *Method* explains how to confirm the expected effect. Other project supervision or an index such as

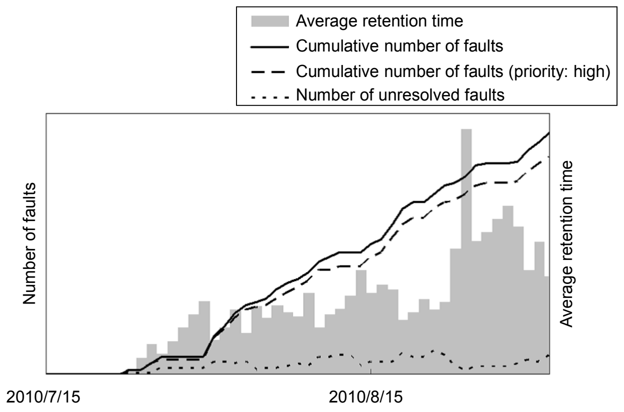
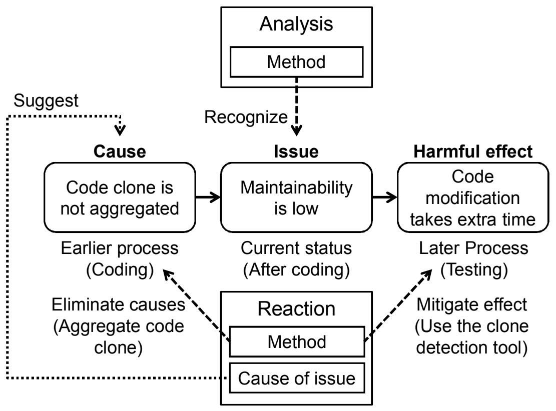


Figure 5. An example of a figure demonstrated on the transformation.



Analysis
Method
Suggest
Recognize
Cause
Issue
Harmful effect
Code clone is not aggregated
Maintainability is low
Code modification takes extra time
Earlier process (Coding)
Current status (After coding)
Later Process (Testing)
Eliminate causes (Aggregate code clone)
Reaction
Method
Cause of issue
Mitigate effect (Use the clone detection tool)

Figure 6. The roles of the analysis and the reaction.

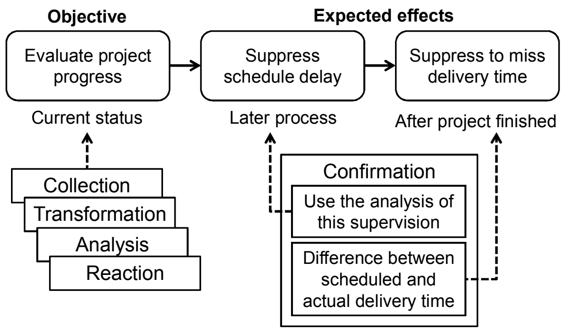


Figure 7. The role of the confirmation.

"difference between scheduled and actual delivery time" in Fig. 7 is used to confirm it. If the period is "in later process," the expected effect is sometimes confirmed by the same method as the analysis (e.g. "Use the analysis of this supervision" in Fig. 7). The method is not written if the expected effect is not confirmed quantitatively. The role of the effect is glue of other expected effects.

One or more of the confirmation is included in an instance of supervision. An example of the confirmation is provided below. The identifier is parenthesized characters at the beginning of the example.

(C1) Period: in later process
Expected effect: suppress schedule delay
Method: use the analysis of this supervision

(C2) Period: after project finished
Expected effect: suppress to miss delivery time
Method: check difference between scheduled and actual delivery time

IV. Comparison of Other Models

Although there are some models which relates to measurement process, they do not just match modeling project supervision. Basili et al. [1] proposed GQM approach. GQM is used to decide which metrics should be measured. At first, a goal which is intended to be achieved through measurement process is set, next, questions which explain how to evaluate goal is set, and metrics are decided based on the questions. GQM is useful to make measurement plan, but it does not express how to do that. Other GQM approach [3] is also different from our model in the points.

GQM+ Strategies [2] is extended model of GQM, and it is proposed to make measurement and improvement activities plan for whole organization. The model includes elements similar to the reaction and the confirmation of our model. However, GQM+ Strategies do not describe detail of them like our model, because the model is not assumed to be applied to in-process supervision. So it does not satisfy Req. 5 and Req. 6 sufficiently.

Kitchenham et al. [20] proposed modeling method of measurement, based on the model which one of the authors proposed [21]. To enhance reliability of dataset, they focused on data structure and storing data, and defined some elements such as data type, range, counting rule, and so on. Namely, their model mainly covers data collection.

ISO/IEC 15939 [15] defines the measurement information model. It mainly covers do and check phase. However, it does not express effective of the measurement activity. ISO/IEC 15939 defines measurement process based on plan-do-check-act (PDCA) cycle, and mentioned act phase. However, ISO/IEC 15939 does not define elements for addressing issues, and therefore the measurement information model does not include them.

Chirinos et al. [6] proposed the model for software measurement (MOSME) which can express objective, data collection activities, and analysis activities. The model has elements which explain collecting and interpreting data in detail. García et al. [12] proposed software measuring modeling language (SMML), based on researches which some of the authors worked on [11][13]. It includes elements which are used to illustrate objective, data collection activities, and analysis activities. But these models do not include elements like the reaction and the confirmation of our model. Hence, these models do not fit well for modeling project supervision. Other models [22][24] are also different from our model in the points.

Table II shows whether or not our model and these models satisfy requirements of a model for project supervision explained in section II (Similar comparison was also done in [6] to clarify differences of past researches). In the table, "Yes" of each cell means a model written in the column satisfies a requirement written in the row, and "No" means not satisfy. Except for our model, any model does not satisfy all requirements completely, and especially, Req. 5 and Req. 6 are not satisfied sufficiently by them. This means without our model, though combination of other models cannot satisfy Req. 5 and Req. 6 completely. To satisfy Req. 5 and Req. 6, a model has to present relationships among

TABLE II. Comparison of Models Related to Measurement Process.

Requirement	Our model	GQM approach [1]	GQM+ Strategies [2]	Kitchenham et al. [20]	ISO/IEC 15939 [15]	MOSME [6]	SMML [12]
Req. 1. Objective	Yes	Yes	Yes	No	Yes	No	No
Req. 2. Collection	Yes	No	No	Yes	Yes	Yes	Yes
Req. 3. Transformation	Yes	No	Partially yes	Yes	Yes	Yes	Yes
Req. 4. Analysis	Yes	No	Partially yes	No	Yes	Yes	Yes
Req. 5. Reaction	Yes	No	Partially yes	No	No	No	No
Req. 6. Confirmation	Yes	No	Partially yes	No	No	No	No

analysis, reaction, and confirmation, in addition to describing them. Our model can present the relationships with FTA diagram, and describe activities of them with the elements.

There are tools (in-process software engineering measurement and analysis systems [17]) which help collecting software metrics and analyzing them [7][26][27]. Although they are useful for project supervision, using the tool only is not sufficient to perform it. They do not support activities of reaction to issues and confirmation of expected effects. In addition, not all of metrics are automatically collected and analyzed by tools. Therefore, a project supervision plan made by our model is needed, if the tools are used.

The requirements of a model for project supervision are similar to some practices stated in CMMI [4]. For example, performing corrective actions and confirming effects of them are mentioned in Project Monitoring and Control (PMC) process area. Also, clarifying measurement objectives, measures, and analysis procedures are required in Measurement and Analysis (MA) process area. However, CMMI is guidance of process improvement, and therefore it is not used to make a project supervision plan. In contrast, our model may be useful to make process definition based on CMMI.

The concept of the Balanced Scorecard (BSC) [19] is akin to our model. BSC is used to make strategic plan of an organization. In BSC, toward an objective, lagging indicators and leading indicators are set, and target values of them are settled. Also, methods to achieve the values (initiatives) are decided. The initiative is somewhat analogous to corrective action, and lagging indicator is similar to confirming effects. But BSC is not suitable for describing a project supervision plan, because it cannot specify how to perform supervision activities.

V. Conclusions

We propose the model of project supervision. Project supervision is performed by sharing software metrics data with a purchaser and a developer, and based on the results, addressing issues is conducted. Although project supervision is expected to suppress project failure, there was no appropriate model to describe it. We specify six requirements for the model, and we proposed new model which consists of six elements corresponds to the requirements. Our model is useful for planning project supervision more rigorously, and a purchaser and a developer can agree the plan more smoothly. Compared to other measurement models, our model is most fitted to project supervision.

As future work, we will make a catalog of project supervision based on our model. Also, we will collect some case studies to evaluate our model. Some concrete guidance when to use our model is needed to diffuse it.

Acknowledgment

This work is being conducted as a part of the StagE project, The Development of Next-Generation IT Infrastructure, supported by the Ministry of Education, Culture, Sports, Science and Technology.

References

[1] V. Basili, and H. Rombach, “The TAME project: towards improvement-oriented softwareenvironments,” IEEE Transactions on Software Engineering, vol. 14, no. 6, pp. 758-773, 1988.

[2] V. Basili, J. Heidrich, M. Lindvall, J. Münch, M. Regardie, and A. Trendowicz, “GQM+ Strategies - Aligning Business Strategies with Software Measurement,” Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM), pp.488-490, Sep. 2007.

[3] L. Briand, S. Morasca, and V. Basili, “An Operational Process for Goal-Driven Definition of Measures,” IEEE Transactions on Software Engineering, vol. 28, no. 12, pp. 1106-1125, 2002.

[4] Carnegie Mellon Software Engineering Institute, CMMI for Development, version 1.3, Carnegie Mellon Software Engineering Institute, no. CMU/SEI-2010-TR-033, 2010, http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

[5] P. Chen, “The entity-relationship model - toward a unified view of data,” ACM Transactions on Database Systems (TODS), vol. 1, no. 1, pp. 9-36, 1976.

[6] L. Chirinos, F. Losavio, and J. Bøegh, “Characterizing a data model for software measurement,” Journal of Systems and Software, vol. 74 no. 2, pp. 207-226, 2005.

[7] M. Ciolkowski, J. Heidrich, F. Simon, and M. Radicke, “Empirical results from using custom-made software project control centers in industrial environments,” Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 243-252, Oct. 2008.

[8] C. Ericson, Hazard Analysis Techniques for System Safety, Wiley-Interscience, 2005.

[9] Food and Agriculture Organization of the United Nations, Food Quality and Safety Systems: A Training Manual on Food Hygiene and the Hazard Analysis and Critical Control Point (Haccp) System, Food and Agriculture Organization of the United Nations, 1998, http://www.fao.org/docrep/W8088E/W8088E00.htm.

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

[11] F. García, M. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero, “Towards a consistent terminology for software measurement,” Information and Software Technology, vol. 48, no. 8, pp. 631-644, 2006.

[12] F. García, F. Ruiz, C. Calero, M. Bertoa, A. Vallecillo, B. Mora, and M. Piattini, “Effective use of ontologies in software measurement,” The Knowledge Engineering Review, vol. 24 , no. 1, pp. 23-40, 2009.

[13] F. García, M. Serrano, J. Cruz-Lemus, F. Ruiz, and M. Piattini, “Managing software process measurement: A metamodel-based approach,” Information Sciences: an International Journal, vol. 177, no. 12, pp. 2570-2586, 2007.

[14] K. Inoue, “Software Tag for Traceability and Transparency of Maintenance,” Proc. International Conference on Software Maintenance (ICSM 2008), pp. 476-477, Oct. 2008.

[15] International Organization for Standardization/International Electrotechnical Commission, ISO/IEC 15939:2007 - Systems and software engineering - Measurement process, International Organization for Standardization/International Electrotechnical Commission, 2007.

[16] International Organization for Standardization, ISO 9001:2008 - Quality management systems - Requirements, International Organization for Standardization, 2008.

[17] P. Johnson, “Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System,” Proc. International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 81-90, Sep. 2007.

[18] T. Kamiya, S. Kusumoto, and K. Inoue, “CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source

Code," IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654-670, 2002.

[19] R. Kaplan, and D. Norton, The Balanced Scorecard: Translating Strategy into Action, Harvard Business Press, 1996.

[20] B. Kitchenham, R. Hughes, and S. Linkman, "Modeling Software Measurement Data," IEEE Transactions on Software Engineering, vol. 27, no. 9, pp. 788-804, 2001.

[21] B. Kitchenham, S. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," IEEE Transactions on Software Engineering, vol. 21, no. 12, pp. 929-944, 1995.

[22] J. Lawler, B. Kitchenham, "Measurement Modeling Technology," IEEE Software, vol. 20, no. 3, pp. 68-75, 2003.

[23] Y. Higo, Libra, http://sel.ist.osaka-u.ac.jp/icca/libra-e.html.

[24] Department of Defense and US Army, Practical Software and Systems Measurement: A Foundation for Objective Project Management, v. 4.0b, 2000, http://www.psmsc.com/PSMGuide.asp.

[25] Object Management Group/Business Process Management Initiative, Business Process Model and Notation (BPMN), version 1.2, Object Management Group/Business Process Management Initiative, no. formal/2009-01-03, 2008, http://www.omg.org/spec/BPMN/1.2.

[26] M. Ohira, R. Yokomori, M. Sakai, K. Matsumoto, K. Inoue, and K. Torii, "Empirical project monitor: a tool for mining multiple project data," Proc. International Workshop on Mining Software Repositories (MSR), pp. 42-46, May 2004.

[27] Quantitative Software Management. Inc., SLIM-Control, http://www.qsm.com/tools/slim-control

[28] World Health Organization, Good Manufacturing Practices and Inspection (Quality Assurance of Pharmaceuticals), World Health Organization, 2007, http://www.who.int/entity/medicines/areas/quality_safety/quality_assurance/QualityAssurancePharmVol2.pdf.