Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 72 (2015) 40 – 48

Procedia Computer Science

The Third Information Systems International Conference

Capability Maturity Model Integration (CMMI) for Optimizing Object-Oriented Analysis and Design (OOAD)

Evi Septiana Pane[a], Riyanarto Sarno[b] *

[a]*Badiklat Industry of Surabaya - Ministry of Industry, Surabaya, Indonesia*
[b] *Informatics Engineering - Sepuluh Nopember Institute of Technology, Surabaya, Indonesia*

Abstract

The current popular method in analysis and design phase is object-oriented analysis design (OOAD). The OOAD method aims to identify the objects which are responsible for their own. Mostly, the measurements of OOAD results are using verification and validation technique. A more demanding approach is expected from the OOAD method to improve software development process. To address this issue, this paper use Capability Maturity Model Integration (CMMI) as integrated software process improvement standard and approach. The purpose of CMMI is to improve process in creating product or services within organization. Therefore, the idea of improving process using CMMI will be applied for optimizing OOAD.

The proposed method in this study is by constructing checklist criteria which already comply with requirement development process area in CMMI. Each criterion will be given certain score, then the score will summarize into total number from the assessment towards the Unified Modeling Languange (UML) Diagrams.

The result obtained score of 185 for all criteria in the checklist and achieved 77,08% of CMMI level implementation, which categorized into Large Implementation (LI). The score also means that the UML diagrams are quite good and could be delivered through the next steps of software development life cycle.

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)

Keywords : Capability Maturity Model Integration (CMMI); object-oriented analysis and design (OOAD); optimizing OOAD; UML Diagrams

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)
doi:10.1016/j.procs.2015.12.103

1. Introduction

Software quality is largely determined by software developments process [2]. The results of each step in software development life cycle must be assured as to have good quality to contribute in software product quality. As preliminary steps in software development life cycle, analysis and design phase plays important role in defining software scope from the perspective of stakeholders and translating them to the engineer using visual artifacts.

In this study, the method used in analysis and design phase is object-oriented analysis and design (OOAD). The OOAD is a powerful method for analyzing and capturing user requirements as well as designing applications by performing visual artifacts to obtain better software quality throughout software development life cycle. When the OOAD is used properly, it will improve the maintenance, reusability, and modifiability of the software [6].

As mention above, OOAD is commonly used in analysis and design phase of software development, therefore some approach for verification and validation of OOAD results is apply to ensure that OOAD results is already compliance with user needs. There are several techniques for verification and validation of UML diagrams (OOAD Results) that are used to perform detailed examination of the design or requirement analysis developed by an analyst. The most popular technique is the use of checklist that present some list of verification derived from software quality standard which already given.

Some software quality standards is used, namely IEEE Std 830-1998 for Software Requirement Specification. A lot of standards are available during software development process which focusing in every phase of software development cycle, these sometimes creating difficulty to compiled all of those standards for creating checklist in verification and validation of UML Diagrams. To address those issue, this research propose the use of Capability Maturity Model Integration (CMMI) as reference model in developing the checklist. CMMI is firstly used as reference model for an organization to improving process performance. The CMMI model is a set of best practices for organization to improve performance in terms of software development process. In addition, CMMI is also used as comparison or complementary model to other standards and method such as SCRUM [1], quality management standard ISO 9001:2000 [7], and Agile framework [4]. As the increasing of benefits obtained by organization after using CMMI [3], more approach is expected from CMMI for optimizing software development process. Hence, this research aim to to optimize object-oriented analysis and design results by creating a checklist which derive from CMMI.

There are two issue to address in this research. First, indicate whether CMMI can be use to optimize OOAD, particularly in UML Diagrams. Second, explain how CMMI is use to optimize OOAD. This research is then applied to the development of training information system in Badiklat Industry of Surabaya.

This paper begins with introduction to the background of why using CMMI to optimize OOAD, that focuses on creating the mapping from CMMI to develop checklist that used in verification and validation of UML Diagrams. Then a more detailed definition concerning the proposed method of OOAD and CMMI model for optimizing OOAD is given. In section three, the evaluation of the OOAD results and optimized OOAD results using CMMI is elaborated. The conclusions are given in the last section.

2. The Proposed Method

2.1. Object Oriented Analysis and Design (OOAD)

Generally, determination of system requirement and identification of classes and their relationship to other classes in the problem domain are the main activity during object-oriented analysis phase of software development. In object-oriented analysis, there are three key-steps to be performed:

1. Identify the Object (actors, functionalities, system entities),

2. Ilustrate how the Object interrelate (use case modeling),
3. Specify the attributes and behavior of the objects (use case detailed description).

Common models used in OOA are use cases. Use case diagrams—deliver the complete view and scope of functionality. The use cases in these diagrams have the behavioral (or functional) explanations of each actor in systems [8].

The first stage of OOA activity encompass identifying objects and classes , which cover people, places, entities, organizations, concepts, and events. Subsequently, relation among entities are documented in form of entity relationship diagrams (ERD). The attributes and services of each class are then identified and documented in class templates. Meanwhile, the aim of object-oriented design is to devise the classes which determined during the analysis phase and also creating the user interface prototype. Similarly in analysis phase, there are three -main steps to be performed during object-oriented design, there are [9]:

1. Arrange interaction diagrams for each scenario (activity diagrams as an output),
2. Build the detailed class diagram,
3. Proceed to the detailed design.

The OOAD come out as a structure for design artifacts, i.e. software coverage and objective, conceptual design, physical design and implementation. To capturing the artifacts of OOAD, analyst commonly utilize Unified Modeling Language (UML) as a graphical language. Hence, some UML Diagrams is constructed as the results from the OOAD; there are use case diagrams, class diagrams and activity diagrams.

2.2. CMMI for Optimizing OOAD

The CMMI is a framework for advancing business process. The CMMI models give direction that can be is used while developing processes. There are currently three sort of CMMI, the most noted one is CMMI for Development. The two other kind are CMMI for Acquisition and CMMI for Services. Each type has different content that targets improvements in particular areas.

The CMMI for Development is a directive model that involves activities for developing both products and services. The CMMI-DEV is arranged based on three concepts: process area (PA), specific goals (SG) and specific practices (SP). There are 16 process areas (PA) contained in CMMI. These process areas encompass basic concepts that are fundamental to process improvement in any area of interest. Process Area definition is a series of related practices in an area that, when implemented simultaneously, fulfill a set of goals considered important for making improvement in that area [10].

The CMMI-DEV use levels to describe the development path that is offered for an organization that wants to improve the process in order to develop its products or services. Maturity levels apply to an organization's process improvement achievement across multiple process areas. These levels are a means to improve the processes corresponding to a given set of process areas (i.e., maturity level). The five maturity levels are numbered 1 through 5 [10].

Process area which is used in engineering process of product or services in the CMMI-DEV is Requirement Development (RD) Process. This process area is meant to be used in engineering process area at maturity level 3. The purpose of Requirements Development (RD) is to elicit, analyze, and establish customer, product, and product component requirements. This process area consists of three Specific Goals (SG) which describe into several Specific Practices (SP) and five Generic Goals (GG) which are explained into some Generic Practices (GP). Table 1. shows detailed information about the content summary of requirement development process area.

Table 1. Specific Goals and Practices Summary

SG 1 Develop Customer Requirements	
SP 1.1	Elicit Needs
SP 1.2	Transform Stakeholder Needs into Customer Requirements
SG 2 Develop Product Requirements	
SP 2.1	Establish Product and Product Component Requirements
SP 2.2	Allocate Product Component Requirements
SP 2.3	Identify Interface Requirements
SG 3 Analyze and Validate Requirements	
SG 3.1	Establish Operational Concepts and Scenarios
SG 3.2	Establish a Definition of Required Functionality and Quality Attributes
SG 3.3	Analyze Requirements
SG 3.4	Analyze Requirements to Achieve Balance
SG 3.5	Validate Requirements

The idea to optimizing the OOAD during software development life cycle using requirement development process area is derived from the basic concept of the CMMI to improve process in an organization, particularly in this case is in improving software development process. By constructing a checklist that interprets from specific goals and specific practices in requirement development process area which given in table I above, the UML Diagrams resulting from the OOAD are assessed. The checklist is also adapted with the general rules of creating UML Diagrams. Table 2. shows the CMMI Checklist criteria for UML Diagrams.

Table 2. Mapping CMMI Process Area to Checklist Criteria for UML Diagrams

No.	Criteria	CMMI Process Area
	General	
1.	Unique name	SG 1 SP 1.1. SG 2 SP 2.1.
2.	Reflects the functional role and purpose of the use case	SG 1 SP 1.2. SG 2 SP 2.1.
3.	Includes all relevant actors	SG 2 SP 2.1.
4.	Avoids ambiguities, omissions, and impreciseness	SG 1 SP 1.1. SG 3 SP 3.5.
	Pre-Conditions	
1.	Explicit	SG 2 SP 2.1.
2.	Describes the state of the system when the use case begins, including data that must exist as well as data that must not exist	SG 3 SP 3.1 , SP 3.2
3.	Captures a valid business situation	SG 3 SP 3.5.
	Basics Flows	
1.	Consistent and compliant with relevant business rules	SG 3 SP 3.3
2.	Includes only information that is relevant to the use case	SG 1 SP 1.1. SG 3 SP 3.1, SP 3.2, SP 3.3
3.	Includes no references to other use cases that include this use case	SG 3 SP 3.1, SP 3.2, SP 3.3
4.	Includes events in the correct order	SG 3 SP 3.1, SP 3.2, SP 3.3, SP 3.4.
5.	References named alternate and exception flows at the appropriate points	SG 3 SP 3.3
6.	References non-functional requirements that are specific to the use case, as applicable	SG 3 SP 3.2.
7.	Uses actor names instead of pronouns	SG 3 SP 3.4.
8.	Avoids the use of adverbs, adjectives, pronouns, synonyms, and negatives	SG 3 SP 3.4.
9.	Uses the present tense throughout	SG 3 SP 3.4, SP 3.5.
10.	Avoids the use of design terms, such as user interface design or database design terms, including	SG 2 SP 2.2, SP 2.3.

No.	Criteria	CMMI Process Area
	but not limited to: click, button, screen, record, table, database, process, data, information	
	Alternate Flows	
1.	Named	SG 2 SP 2.1.
2.	Comprehensive	SG 2 SP 2.1. SG 3 SP 3.1, SP 3.2, SP 3.3
3.	Correctly state its starting point in the basic flow	SG 3 SP 3.5.
4.	Last step indicates the returning point within the basic flow	SG 3 SP 3.5.
	Post - Conditions	
1.	Explicit	SG 2 SP 2.1.
2.	Captures a valid business situation	SG 3 SP 3.5.
3.	Describes the state of the system when the use case ends, including data that must exist as well as data that must not exist	SG 3 SP 3.2, SP 3.3

The checklist is then applied to assess the UML Diagrams with certain score. The scoring intervals are from 1 to 10. The least score means that the diagrams do not fulfill the criteria in the checklist, while higher score means that the diagrams fulfill the criteria in the checklist. By looking to the diagrams (use case description, class diagram and activity diagram), analyst then quantify based on the criteria in the checklist. After quantify the diagrams, the summary of the score criteria is compared to maximum score of the checklist compliance, which is 240. The percentage of the score is then classified into four categories of implementation like shown in Table III. Then, we can conclude whether the diagram models are quite good to be implemented in the next stage or must be repaired before going to the next stages.

Diagrams resulting from the OOAD can be optimized by using this checklist to assess the implementation of the CMMI concept of requirement development process area. This approach could be used as anticipation method to reduce the cost of failure in the next stage causes by 'error' in UML Diagrams.

Table 3. CMMI Implementation Level Category

Label	Meaning	Score (%)
Fully Implemented (FI)	- Direct artifact is present and judged appropriate. - At least 1 direct artifact and/or affirmation exist to confirm implementation. - No substantial weakness noted	76 – 100
Largely Implemented (LI)	- Direct artifact is present and judged appropriate. - At least 1 direct artifact and/or affirmation exist to confirm implementation. - One or more weaknesses documented	51 – 75
Partially Implemented (PI)	- Direct artifact is absent or judged to be inadequate. - Artifacts/affirmation suggests that only some aspects of the practice are implemented. - Documented weaknesses which impact goals	26 – 50
Not Implemented (NI)	Any situation not covered above,	0 – 25

3. Evaluation

The outcomes from this study are divided in two parts. First, the result from object-oriented analysis and design phase of training information system. Second, the result of optimizing the CMMI checklist for UML Diagrams as an output from object-oriented analysis and design phase.

3.1. Object Oriented Analysis and Design Results

The Object-oriented Analysis (OOA) phase results in three problems areas which will be the focus of this information system. They are planning, perform and evaluate the training. Those problem areas are then translated into functional requirement and non-functional requirement. The functional requirements draws five functionalities: managing training program, managing training participants' data, managing participants' training certificates, managing the execution of training program, and managing the training evaluation process. The non-functional requirements of the training information systems are expected to address performance, security, reliability, and modifiability.

The OOA phase also determines the actors who are involved in information system. In the case of training information system, there are 3 primary actors being involved, i.e.: training plan administrator, training evaluation administrator, and training execution administrator, which is shown in Figure 1 below.

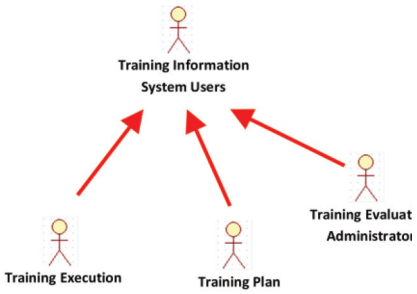
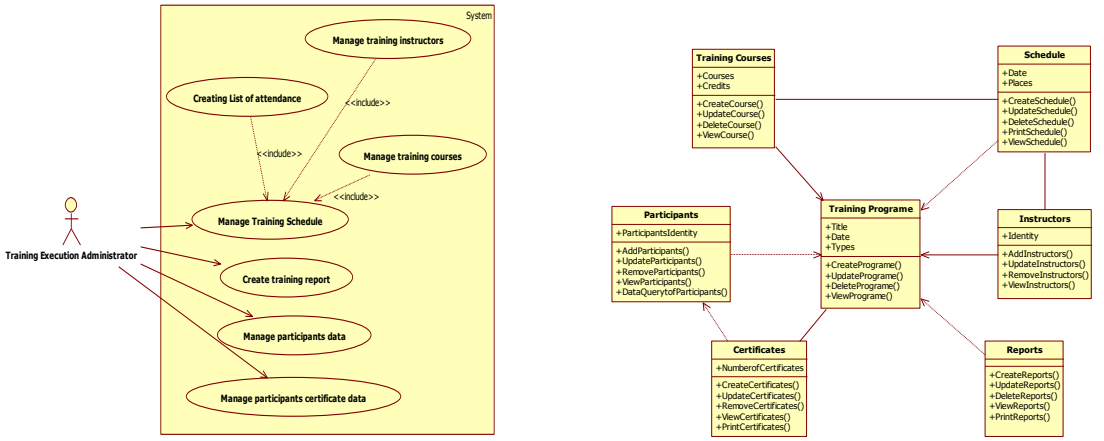


Fig. 1. Primary Actors in Systems

The interactions between objects are represented using use case diagrams. The total number of use case activities in this system is 14 use cases that comprises of 7 main use cases and 7 descendent use cases. The use case activities given in use case diagrams are translated from functional requirement that already determined in earlier phase. Figure 2(a) shows partly capture of use case activity from entire use case diagrams by the perspective of training execution administrator actor.

The object-oriented design (OOD) results in UML diagrams (i.e. class diagrams, activity diagrams, and use case detailed descriptions) for each functionality. Figure 2(b) shows class diagram which is drawn from use case diagram from user training execution administrator's view. While Figure 3 shows example of activity diagram for managing training schedule functionality.



2(a) 2(b)

Fig. 2. (a) Use Case Diagram of Systems for Actor Training Execution Administrators; (b) Class Diagram of Systems for Actor Training Execution Administrators

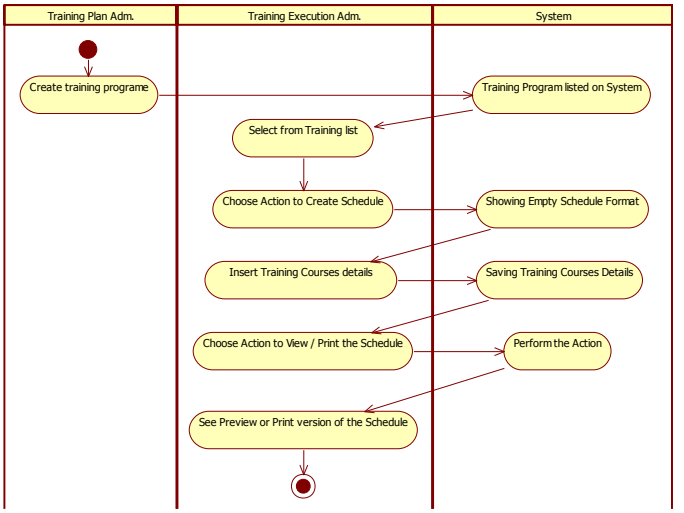


Fig. 3. Example of Activity Diagram in Use Case Manage Training Schedule

3.2. OOAD Results after Assessed Using The CMMI Checklist

Based on the checklist given in proposed method, we quantify the score for entire UML diagrams in every functionality of system. The score results from the checklist will lead to conclusion. From the scoring, it can be obtained whether the UML Diagrams comply with CMMI model or not. Table 4 show one of the scoring examples between all functionalities which the system has. The scoring was doing by an analyst.

Table 4. CMMI Scoring Results on UML Diagrams of Funcionality : Manage Training Schedule Level Category

No.	Criteria	Score

No.	Criteria	Score
	General	
1.	Unique name	10
2.	Reflects the functional role and purpose of the use case	9
3.	Includes all relevant actors	9
4.	Avoids ambiguities, omissions, and impreciseness	8
	Pre-Conditions	
1.	Explicit	9
2.	Describes the state of the system when the use case begins, including data that must exist as well as data that must not exist	8
3.	Captures a valid business situation	9
	Basics Flows	
1.	Consistent and compliant with relevant business rules	9
2.	Includes only information that is relevant to the use case	8
3.	Includes no references to other use cases that include this use case	7
4.	Includes events in the correct order	9
5.	References named alternate and exception flows at the appropriate points	5
6.	References non-functional requirements that are specific to the use case, as applicable	3
7.	Uses actor names instead of pronouns	4
8.	Avoids the use of adverbs, adjectives, pronouns, synonyms, and negatives	7
9.	Uses the present tense throughout	8
10.	Avoids the use of design terms, such as user interface design or database design terms, including but not limited to: click, button, screen, record, table, database, process, data, information	8
	Alternate Flows	
1.	Named	8
2.	Comprehensive	7
3.	Correctly state its starting point in the basic flow	7
4.	Last step indicates the returning point within the basic flow	7
	Post – Conditions	
1.	Explicit	10
2.	Captures a valid business situation	8
3.	Describes the state of the system when the use case ends, including data that must exist as well as data that must not exist	8
	TOTAL SCORE	185

The fulfillment of UML diagram towards CMMI model reference which quantified using CMMI checklist obtained total number of 185, those number means that the UML diagram fulfill 77,08% of CMMI checklist and categorized in Largely Implemented (LI) area. These result category means that most of entire UML Diagrams from OOAD phase comply with the criteria in CMMI in Requirement Development Process Area. Then, the modeling diagrams may be used in the next stages of software developments with some improvement in several criteria from the checklist.

Compared with normal process of creating UML Diagrams in OOAD, this approach gives addition to verify the completeness of the UML Diagrams that refers to the CMMI-DEV model. This verification can also be rated as optimizing the OOAD for the next phase in software development life cycle.

4. Conclusion

Object-oriented analysis and design method plays important role in determining the quality of software development process. Optimizing object-oriented analysis and design results by using the CMMI is relatively new research area in software process improvement. In this paper, carrying out the CMMI to creating the checklist in verification and validation of OOAD results is proposed first, and then how the CMMI checklist being use to optimize the UML Diagrams is discuss. Applying the CMMI checklist to quantify whether the OOAD results are quite good can reduce organization cost of failure in the next stages of software development process.

In the coming works, the CMMI should be more elaborate in another technique of verification and validation of OOAD results. Other challenging topic in OOAD is how to address changes in environment in organization during software development process.

References

[1] C.R. Jakobsen, J. Sutherland, “Scrum and CMMI Going from Good to Great”, Agile Conference, IEEE, pp. 333-337, Chicago, Auguts, 2009.

[2] D. O’Neill, “Extending the Range of Value of CMMI to a new Normal”, CrossTalk, January/February, 2012.

[3] D.R. Goldenson, D.L. Gibson, and R.W. Ferguson, “Evidence about the Benefits of CMMI”, Software Engineering Institute, Carnegie Mellon University, SEPG 2004.

[4] H. Glazer, J. Dalton, D. Anderson, M. D. Konrad, and S. Shrum, “CMMI or Agile : Why Not Embrace Both !”, Technical Report, Software Engineering Institute, Carnegie Mellon University, 2008.

[5] J. D, McKeen, “Successful development strategies for business application systems", MIS Quart., 7: 47, 1987.

[6] M.J. Quillin, “Object Oriented Analysis and Design”, University of Missouri - St. Louis, November 25, 2001.

[7] M.T. Baldassarre, D.Caivano, D.J. Pino, M. Piattini, and G. Visaggio, “Harmonization of ISO/IEC 9001:2000 and CMMI-DEV : from a theoretical comparison to a real case application”, Software Quality Journal, Springer, Vol. 20, Issue 2, pp 309-335, June, 2012.

[8] S. Meena and R. Vishwakarma, “CMMI Based Software Metric for OOAD”, International Journal of Programming Languages and Applications (IJPLA) Vol.3, No. 1, January, 2013.

[9] S. R. Schach, “Object-Oriented and Classical Software Engineering”, McGraw-Hill, New York, 2005.

[10] SEI, “CMMI® for Development,Version 1.3”, CMMI Product Team, Software Engineering Institute, Pittsburgh, November 2010