Toward mature method engineering by using CMMI

Nima Farsadkia, Hassan Rashidi

(Qazvin Azad University, Qazvin, Iran)

Abstract—**this article is related to the area of situational method engineering (SME). In this domain, approaches are developed accordingly to specific project and/or organization specifications. We propose to adapt particular part an existing method construction process, namely the assembly-based with a virtual reference model (WBS model) that extract from Capability Maturity Model Integration (CMMI). Our proposal is to offer a better performance in the retrieval of similar chunks by the introduction of multi criteria (less conflict & high adjusting with CMM standard) techniques.**

***Keywords*— Method Engineering, Method Chunk, CMMI Reference Model, CMMI WBS, Cosine Similarity Measure**

I. INTRODUCTION

Theoretical research in the field of software development methodologies (SDM) shows that use of SDM should improve the efficiency of the development team and the quality of the developed product (1). With the intention o realize improving SDM, we need to have a suitable and qualifiable way to construct methodologies that adapted and be compatible with organizations and/or its projects specific needs and situations. Constructing such a methodology discussed in Method Engineering (ME) and Situational Method Engineering (SME) literature. ME and SME focus on formalizing the use of methods for systems development. The broader term, method engineering, is defined as the engineering discipline to design, construct and adapt methods, techniques and tools for systems development, a definition analogous to the IEEE definition of software engineering (2). A major component of ME is situational method engineering, which encompasses all aspects of creating a development method for a specific situation. Anyway main question that concerned us is:

"How we can ensure that a constructed method has expected quality for an organization and/or its projects?"

Quality can reach by several aspects: such 1) using method architectures and standard architectural styles 2) measure coverage of functional requirements of SDM by constructed method, and so on. But these solutions have an affinity to method engineer knowledge and his/her experience about decomposing of ready-made methodologies to fragments (or chunks), choosing appropriate fragments (or chunks) from method base and using assembly techniques to configure new methods. Deficiency in use ME concepts in any stage and lack experience of method engineer may be cause defect result. We propose using a reference model of best practices that supervise method constructing process to achieve suitable products (methods). This reference model obtains from SEI's Capability Maturity Model Integration (CMMI).

CMMI defines practices that businesses have implemented on their way to success. Practices cover topics that include collecting and managing requirements, formal decision making, measuring performance, planning work, handling risks, and more (3). CMMI is not a process or a process framework in itself, but contains a process reference model used to perform process assessments (4). Therefore this model usage will cause to achieve high maturity and so on high quality in ME results. To perform this idea we define two equivalent structures to constructing method at hand and CMMI based reference model and analyze pair affects for any fragment chose by critical region method. Then when in ME process a fragment was chosen, consider that selected fragment cause to higher maturity of result method and has lowest defects and mutual exclusions. The paper is thus structured as follows: The next section, 2, gives an overview of the ME and SME literature, section 3 presents CMMI and related concepts, next section deal about our twofold proposal (Internal conflicts and External similarity measure), finally in section 5 an example is discussed

II. METHOD ENGINEERING AND SITUATIONAL METHOD ENGINEERING

Method Engineering (ME) was introduced by (5) and then, more recently, by (6) who named it methodology engineering; but (7) and (2) strongly recommend changing this to method engineering, a term that has been generally accepted. Brinkkemper's (2) definition of method engineering is useful here: *"Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems."* When applied to a particular situational context, it is often referred to as *"situational method engineering"* or SME. Interestingly, (8) equates the ME approach to an *"ad hoc"* approach in that the correct meaning of ad hoc is *"suited to purpose"* or "*tailored to the problem at hand"*. Method engineering focuses not on the acquisition of a ready-made method from some supplier but on the in-house construction of an organization-specific or project-specific methodological approach. This construction is accomplished by selecting pieces of method (method fragments or method chunks) that have been already created and stored in a repository or method base. The suitability of the fragments for adding to a method base requires appropriate coherency and granularity (9). To facilitate later retrieval, it is also important that the fragments suitable for storage are documented accurately. In (10) and ISO/IEC 24744 suggest several facilities for formalize, store, retrieve and manipulate fragments.

After method fragments extractions (11), (12), describe a generic *"modular method meta-model"* that provides the ability to represent any method by an assembly of (reusable) method chunks. This process model includes three kinds of SME approaches namely Assembly-based, Extension-based and Paradigm-based and permits the combination of them in a particular SME process. In this paper we use first approach (Assembly-based) to construct appropriate and suitable methods.

In this top-down method construction approach, identification of useful fragments is the remit of the intention select method chunks in the assembly-based process model of figure 1.

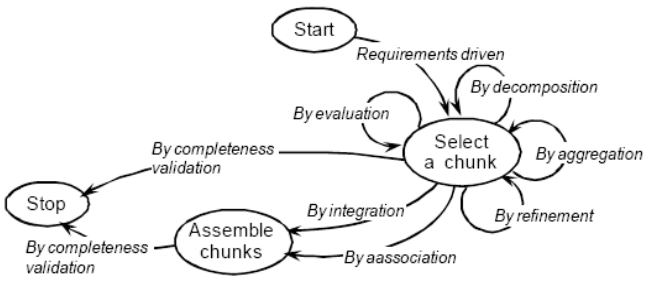

FIGURE 1 - ASSEMBLY-BASED PROCESS MODEL FOR SME (11)

For each retrieved chunk, its potential usefulness is first evaluated (evaluation strategy) – this can be done using similarity measures as described by (11) and extended by (13), who describes three kinds of similarity: 1) The number of common aspects based on *"User Situation"* and *"Reuse Context"*, 2) The forbidden aspects of *"User Situation"* and *"Reuse Context"* and (3) the number of necessary aspects in the *"User Situation"*.

When necessary, refinement of the chunk may be undertaken using one of three further strategies (11):

- Decomposition strategy – where the chunk is a compound one containing parts not needed for the current method construction,
- Aggregation strategy – when the chunk only covers the requirements partially,
- Refinement strategy – suggests seeking another chunk with a richer set of guidelines than the current selection.

The meta-knowledge stored with the method chunks is highly relevant in ensuring a contextual retrieval. Suggestions for an appropriate query language are given in (14). The modeling language, MEL, proposed by (14) also contains a portion useful for identifying and removing method chunks from the database.

III. CAPABILITY MATURITY MODEL INTEGRATION

A Capability Maturity Model (CMM), including CMMI contains the essential elements of effective processes. A focus on process provides the infrastructure and stability necessary to deal with an ever-changing world and to maximize the productivity of people and the use of technology to be competitive (3). Process helps an organization's workforce to meet business objectives by helping them to work smarter, not harder, and with improved consistency. Effective processes also provide a vehicle for introducing and using new technology in a way that best meets the business objectives of the organization.

In the 1930s, Walter Shewhart began work in process improvement with his principles of statistical quality control (16). These principles were refined by W. Edwards Deming (17), Phillip Crosby (18), and Joseph Juran (19). Watts Humphrey, Ron Radice, and others extended these principles further and began

applying them to software in their work at IBM and the SEI (20). Humphrey's book, Managing the Software Process, provides a description of the basic principles and concepts on which many of the Capability Maturity Models (CMMs) are based. CMMs focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness.

Today, CMMI is an application of the principles introduced almost a century ago to this never-ending cycle of process improvement. The value of this process improvement approach has been confirmed over time. Organizations have experienced increased productivity and quality, improved cycle time, and more accurate and predictable schedules and budgets (21). In the CMM, five maturity levels are distinguished: 1) initial, in which capability is characteristic of individuals, not organizations or methods, 2) repeatable, in which project planning is stable and earlier success can be repeated, 3) defined,

TABLE 1- CAPABILITY MATURITY MODEL LEVEL AND PA'S

Level	Focus	Process Areas	Result
5 Optimizing	*Continuous process improvement*	Organizational Innovation & Deployment Causal Analysis and Resolution	Productivity & Quality
4 Quantitatively Managed	*Quantitative management*	Organizational Process Performance Quantitative Project Management	
3 Defined	*Process standardization*	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Risk Management Decision Analysis and Resolution	
2 Managed	*Basic project management*	Requirements Management Project Planning Project Monitoring & Control Supplier Agreement Management Measurement and Analysis Process & Product Quality Assurance Configuration Management	
1 Initial	*Competent people and heroics*		

in which project teams tailor a method to their own project-specific method, 4) managed, in which information system engineering projects are quantifiable and predictable, and 5) optimizing, which can be summarized as continuously improving. Also CMMI-dev contains 22 process areas. Of those process areas, 16 are core process areas, 1 is a shared process area, and 5 are development specific process areas.

IV. SOLUTION: METHOD ENGINEERING

In (22) CMMI breaks in to work breakdown structure (WBS) with three levels: 1) policies 2) Procedures 3) Work products.

A. Dual effects and interaction between MCs

We would assume this structure and its elements are as resources that when a chunk is selected from method base consume necessary capabilities from CMMI's WBS. This method cause that with selecting any chunks we can measure total method (ology) maturity (by coverage of more WBS elements) and detect conflicts of chunks that have negate affects in a once element.

Consuming of capability derived:

1) There are relationships between any chunks and CMMI's WBS elements.

2) If two chunks race to affect in one element and theirs affects negate others, may be cause an exclusion situation. For example: existence of chunk1 mandatory a special work product of CMMI's WBS actualizes and then in other whence with existence of chunk2 same work product make forbidden.

TABLE 2- POSSIBLE PAIR AFFECT OF TWO CHUNKS WITHIN A ELEMENT.

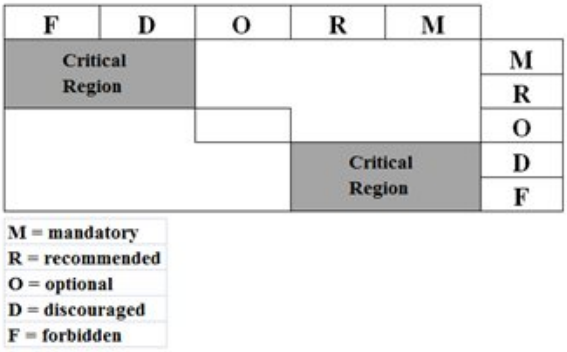

TABLE 2 show 5 probable states of pair effects of two method chunks with a critical region (CMMI's WBS element).

When cause mutual exclusion by negate effect of method chunks some solutions discussed:

- Abort a chunk that has a low suitability and replace with another.

- If second chunk exclude any others, replace that with another.
- If by iteration of prior solutions cannot find suitable way, method architecture or situation must changed and reviewed.

B. Partial and external impacts of MCs on the overall method maturity level

We may find a solution that hasn't any conflicts, but cannot be sure the total quality of result method archived. Because may some necessary key process areas in CMMI model doesn't covered, and choosing of chunks and analyze and prevent of mutual exclusion between them doesn't guaranty coverage of important aspects of method quality that was hidden in CMMI practices and it's WBS. To aim that our result method cover best practices of reference model must declare and use a similarity measure in assembling process to direct this process to best choices. We propose use Cosines similarity measure between two models (n-dimensional equivalent vector of reference model and like vector of constructing model). Using of this measure help us to improve method construction's direction. In some case we didn't need to absolutely obtain to reference model scale, but just enough our method similar and in direct of reference model, because usually reference model to reach out schedule and budget will be.

Cosines similarity measure defines as:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

A, B is n-dimensional vectors of two models. Each element of any vector represents total degree of specific effects of chunks choosing in CMMI's WBS items. For example assume the first element of CMMI's WBS refers to *"Lists of Criteria for Distinguishing Requirements Providers"* work product and we have 4 suitable chunks for a place in our method architecture, The chunk will be selected to lowest angle with reference model (best practices or suitable standards), so we have a two-objective optimization problem: 1) Lowest conflicts between pair chunks 2) Lowest angle between customized CMMI's WBS equivalent vector and like vector for under construction method.

By design a decision tree and its overall traversals; the above optimization issue in small scale projects can be solved. Tree structure is well-defined and easy usage structure and any traversal path from the root to the leaves is a possible solution of the issue. We must compute two numbers for any traversal path: 1) number of solvable or soft[1] conflicts (if insoluble or crisp[2] conflicts exist then path would ignore) between selected chunks on CMMI's WBS's elements, 2) angle between path solution and expected reference model.

The path would select that these numbers are to its least. If project scale is small and desired situations and method architecture is simple, the size of the tree design can be controlled but if situations are some complex and/or method architecture is not well defined may these numbers have mutuality. In the worst case, none of the parameters have not significant advantage against another to be selected; in this case it is better: 1) Method architecture be reviewed 2) Opposite situations are identified and then chunks are select 3) method repository be considered and must new chunks will be made.

May none of these proposals does not resolve the parameters contrasts; In this case we are dealing with a combinatorial issue and can solve it with a well-known optimization algorithm. In simple case, with weighting or outranking these parameters we can convert them to ones and then solve problem simply.

V. EXAMPLE

To illustrate our proposal, we have selected method chunks that deal with information system (IS) security. Five chunks of RE methods designed for analyzing Information System security were identified (23)(30): 1) NFR Framework (23), 2) KAOS (24), 3) Secure Tropos (25), 4) GBRAM (26), and 5) Misuse Cases (27). The comparison of these methods is presented in (28)(30). Within this example, the given project is described by:

- the great influence on the whole organization;
- the need for ensuring the greater progress;
- the organization does not have the experts in this field and does not plan to employ them;
- the need for a better explanation of method chunks and their application.

1 Conflicts between Mandatory or forbidden levels with another level.

2 Conflicts between Mandatory and forbidden levels

The method engineer has chosen three project characteristics and has described the method chunks according to methods properties. Thus, these methods chunks are compared according to six criteria, which concern two groups: project characteristics and proper method characteristics. The first group includes impact, level of innovation, and expertise.

TABLE 3-IS SECURITY CHUNKS EVALUATION (23)

Criteria	NFR Framework	KAOS	Secure Tropos	GBRAM	Misuse Cases
Project Characteristics					
Impact	high	low	high	low	normal
Level of innovation	high	high	low	high	high
Expertise	normal	high	high	normal	low
Method Chunk Characteristics					
Guidance	predefined taxonomy	reuse of generic refinement patterns, heuristics	*No guidance*	documents analysis, heuristics	guidelines
Approach	explanatory	exploratory	systemic	*Not applicable*	explanatory
Formalism	semi-formal	formal	formal	informal	informal

The second group comprises guidance, approach, and formalism. Depending on project description, the method engineer has defined the following preferences rules for these criteria:

- Impact on organization: maximum;
- Level of innovation: maximum;
- Required expertise: minimum;
- Guidance: a predefined taxonomy is better than heuristics, which is better than a simple guidelines;
- Approach: a systemic approach is better than exploratory, which is better than explanatory.
- Formalism: a formal approach is better than semi-formal one, which is better than informal one.

The summary of chunks evaluation is presented in Table 3 (23) .

Now, suppose that in this system we need to cover maintenance property in our architecture, so another chunk in this area must be select. In (29) define three chunks and their comparisons in several concepts: 1) MaSE, 2) Prometheus, and 3) Tropos. Only the chunk that extract from Prometheus Method is support maintenance property. So if because of the need systematically approach we select Tropos chunk, by existence this chunk we didn't cover maintenance feature. Also if we need a chunk to produce guidance for project existence of this chunk avoid that.

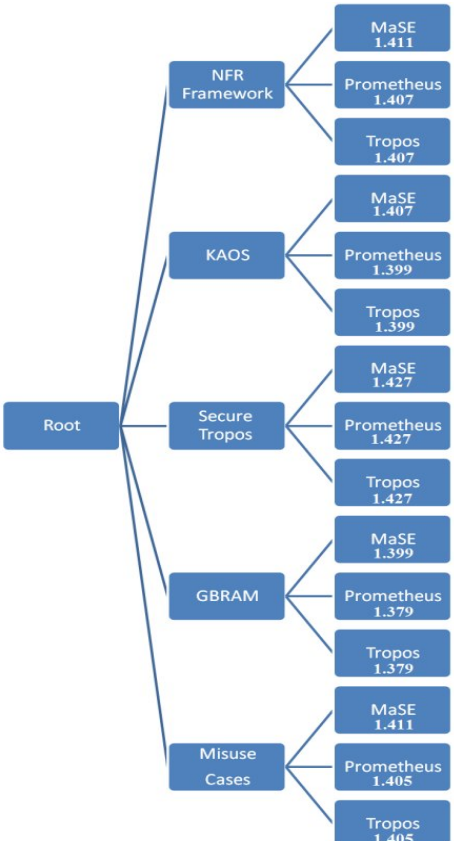

FIGURE 2- THE TREE STRUCTURE TO DECISION MAKING

For second criterion, we design a tree that in first order, security chunk must select and so on maintenance chunk will select, assume that the above-mentioned reference model have preferences rules.

As shown in Figure 2, for any leaf we compute a number that show cosine similarity measure. You see that select a sequence of (GBRAM, Prometheus) or (GBRAM, Tropos) has more similarity with CMMI reference model. But because Tropos has a hard conflict with maintenance feature, second solution will be ignored.

VI. CONCLUSION

We have proposed an adaptation of the existing select and assembly processes with the introduction of MC techniques. The two approaches (exclusion analyze and using CMMI reference model) may be combined within the same method engineering

process as it will offer a more complete guidance to select chunks.

Our objective is twofold. Firstly, we offer the possibility to the method engineer to qualify the method chunks by their correspondence with projects and to choose between similar chunks by an application of MC techniques and exclusion analyze. Secondly, we propose to characterize the project in well-defined structure and analyze effects of chunk select compared with a virtual reference model (VRM). This VRM is not a practical model but a set of rules and standards is based on the CMMI to improve their selection. This typology allows to identify all their critical aspects and to weight them. Within our example, we showed the utility of application of MC techniques.

In near future, our research perspectives include:

- Improve the guidance;
- Using several optimization algorithms to find best solution;
- Improve the typology presented in this paper in order to take into account other critical characteristics considered in CMMI;
- Extend the MC techniques application to the field of System Engineering based on MC techniques chunks using CMMI approach.

REFERENCES

1. Avison D., Fitzgerald G.,. *Information Systems Development: Methodologies, Techniques and Tools.* Third Edition. s.l. : McGraw-Hill Education, 2003.

2. *Method Engineering: Engineering of Information Systems Development Methods and Tools.* Brinkkemper, S. s.l. : Information and Software Technology, 1996, pp. 275-280.

3. SEI. *CMMI® for Development, Version 1.3.* s.l. : SEI, 2010.

4. *Process Construction and Customization.* Henderson-Sellers, Serour, McBride, Gonzalez, Dagher. s.l. : Journal of Universal Computer Science, 2004, Vol. 10, pp. 326-358.

5. *A software development model for method engineering,.* Bergstra, J., Jonkers, H. and Obbink,. North-Holland : Elsevier Science Publishers, 1985.

6. *Methodology Engineering: a Proposal for Situation Specific Methodology Construction.* Kumar, K. and Welke, R.J.,. s.l. : John Wiley & Sons, 1992. Challenges and Strategies for Research in Systems Development. pp. 257-269.

7. *A method engineering approach to information systems development.* van Slooten, K. and Brinkkemper, S.,. North-Holland : Elsevier Science Publishers, 1993. IFIP WG8.1.

8. *Process diversity and a computing old wives'/husbands' tale,.* Glass, R.L.,. s.l. : IEEE Software, 2000.

9. *On the feasibility of situational method engineering.* ter Hofstede, A.H.M. and T.F. Verhoef,. s.l. : Information Systems., 1997, Vol. 22, pp. 401-422.

10. *OLMS – An Object Library Management Support System.* Freeman, C., Henderson-Sellers, B. Sydney : Prentice Hall, 1991, pp. 175-180.

11. *An Assembly Process Model for Method Engineering.* Rolland., J. Ralyte and C. s.l. : Proc. of CAiSE'2001, 2001. Vol. 2068, pp. 267–283.

12. *Guiding the construction ot textual the use case specification.* Rolland C, Ben achour. s.l. : Data and knowlege Engineeing Journal., 1998, pp. 125-160.

13. *Adapting Analysis and Design to Software Context.* Mirbel, I., De Rivieres, V. Proceedings of the 8th International Conference on Object-Oriented Information Systems (OOIS'02),. pp. 223-228.

14. *A Proposal for Context-Specific Method Engineering.* Rolland, C., Prakash, N. Atlanta, USA, : Chapman & Hall, 1996. Proceedings of IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering,. pp. 191-208. In Method Engineering. Principles of Method Construction and Tool Support.

15. *A Method Engineering Language for the Description of Systems Development Methods (Extended Abstract).* Brinkkemper, S., Saeki, M., Harmsen, F. Berlin : Springer-Verlag, 2001. Advanced Information Systems Engineering 13th International Conference, CAiSE 2001. pp. 473-476.

16. Shewhart, W. *Economic Control of Quality of Manufactured Product.* New York : Van Nostrand, 1931.

17. Deming, W. Edwards. *Out of the Crisis.* Cambridge, MA : MIT Press, 1986.

18. Crosby, Philip B. *Quality Is Free: The Art of Making Quality Certain.* New York : McGraw-Hill, 1979.

19. Juran, Joseph M. *Juran on Planning for Quality.* New York : Macmillan, 1989.

20. Humphrey, Watts S. *Managing the Software Process.* Boston : Addison-Wesley, 1989.

21. Gibson, Diane L., Goldenson, Dennis R. and & Kost, Keith. *Performance Results of CMMI-Based Process Improvement (CMU/SEI-2006-TR-004, ADA454687).* Pittsburgh : Software Engineering Institute, Carnegie Mellon University, 2006.

22. F.Rico, david. Software process Improvement Using. [Online] http://davidfrico.com/s-cmmi-wbs.pdf.

23. L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos. *Non-functional requirements in software engineering.* s.l. : Kluwer Academic Publishers, 1999.

24. *Goal-directed Requirements Acquisition, Science of Computer Programming.* A. Dardenne, A. Lamsweerde, and S. Fickas. s.l. : Elsevier, 1993, Vol. 20.

25. *TROPOS: An Agent Oriented Software Development Methodology.* P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. s.l. : Journal of Autonomous Agents and MultiAgent Systems, 2004, Vol. 8, pp. 203-236.

26. *Goal Identification and Refinement in the Specification of Software Based Information Systems.* A.I. Anton. Atlanta, USA : Ph.D. Dissertation, Georgia Institute of Technology, 1997.

27. *Misuse cases help to elicit non-functional requirements.* Alexander, I. s.l. : Computing & Control Engineering Journal, 2003, Vol. 14, pp. 40-45.

28. M. Lassoued and C. Salinesi. *Shall IS Security be Treated Differently in the light of the Open World Assumption? A Literature Review.* Paris : Centre de Recherche en Informatique,, 2006.

29. *Comparing Agent-Oriented Methodologies.* Khanh Hoa Dam, MichaelWinikoff. Berlin : Springer-Verlag, 2004.

30. *Method Chunks Selection by Multicriteria Techniques: an Extension of the Assembly-based Approach.* Kornyshova E., Deneckère R., and Salinesi C. Geneva, Switzerland : Situational Method, 2007.