Towards CMMI-compliant MDD Software Processes

Alexandre M. L. de Vasconcelos
Centro de Informática
Universidade Federal de Pernambuco
Av. Jornalista Anibal Fernandes s/n, 50740-560
Cidade Universitária, Recife-PE, Brazil
e-mail: amlv@cin.ufpe.br

Giovanni Giachetti, Beatriz Marín, Oscar Pastor
Centro de Investigación en Métodos de
Producción de Software - PROS
Universitat Politècnica de València
Camino de Vera s/n, 46022, Valencia, Spain
e-mail: {ggiachetti, bmarin, opastor}@pros.upv.es

***Abstract* — In the last years, Model-Driven Development (MDD) approaches have taken an important role in the quality improvement of software products. These approaches perform the automatic compilation of high-abstraction models to generate the final application code. In this way, MDD approaches aim at reducing development costs as well as increasing productivity, portability, interoperability, and ease of software evolution; i.e., achieving higher product quality. A major obstacle for MDD approaches to be massively adopted by industry is their lack of alignment to well-defined quality models for software processes. We advocate that performing a compliance analysis, based on a software process quality model, is the first step to deal with this obstacle. In this paper, we analyze the degree of compliance of an industrially applied MDD approach with the CMMI-DEV quality model. In particular, we determine those characteristics that meet the technical solution process area of CMMI-DEV and identify improvement opportunities to obtain a proper alignment of the MDD approach with this model.**

Keywords – MDD; OO-Method; CMMI; Software Process Quality; Feature-based Analysis.

I. INTRODUCTION

Developing high quality software has been a continuous concern in the Software Engineering community. To achieve this goal, several software development approaches have emerged. In this context, the Model Driven Development (MDD) approach [1][3][4] has become subject of current research. The main idea behind MDD is the automatic generation of code from models through successive transformation of higher abstraction's level models (problem domain) into more concrete models (solution domain).

The MDD paradigm advocates that the initial software development and the implementation of future changes are all made in the model. In this way, MDD allows lower development costs, and higher productivity, portability, interoperability, and ease of software evolution [5]; i.e., higher software quality.

In parallel to the research on MDD and to the gradual adoption of this approach, many software development organizations are strongly seeking improvement and/or assessment of their software processes on the basis of quality models [6][7]. Such organizations aim to improve the efficiency of their processes and the quality of the products developed by the enactment of these processes as well as to meet market and stakeholders needs. Hence, given the importance of software process quality models, MDD approaches must be compliant with these models to be widely used by software development organizations. Since this challenge has not been properly addressed by any MDD approach yet, further research into this direction is necessary. Thus, we propose the following research question: "*Is it possible to design a MDD process that fully complies with a well-defined software process quality model?*"

We advocate that MDD approaches must be analyzed with regard to software process quality models as a first step towards answering this research question. In this paper, we analyze the compliance of a specific MDD approach, named OO-Method [8], with the Technical Solution (TS) Process Area (PA – a cluster of related practices that, when implemented collectively, satisfies a set of goals for making improvements in an area) of the CMMI-DEV (Capability Maturity Model Integration for Development) software process quality model [9]. This analysis is performed by using an assessment method based on SCAMPI (Standard CMMI Appraisal Method for Process Improvement) [10]. As consequences of this analysis, certain weaknesses in the OO-Method approach have been indicated and adjustments have been proposed to make this MDD approach fully compliant with TS. Thus, the OO-Method approach can be integrated into a complete CMMI-based software process.

OO-Method and CMMI (across the entire paper CMMI and CMMI-DEV are being used as synonyms) have been chosen for the analysis because the former is a MDD approach that has been successfully applied in the software industry [11] and the latter is the most frequently adopted software process quality model [6][7]. TS has been chosen to be analyzed because the main objectives of this PA are the design and the implementation of information system's requirements, which are also the main objectives of MDD.

The contribution of this paper is twofold. First, practitioners can benefit from the analysis by adapting it to detect weaknesses on other MDD approaches in relation to a software process quality model. As a result, they can decide whether adopt a specific approach (although having to modify it) or discard it and adopt another one. Second, this type of analysis can be useful in academia for identifying room for improvement in existing or new MDD approaches, and therefore, for discovering further research areas. By using this analysis as reference, other MDD approaches can

 ISBN: 978-1-61208-165-6

be analyzed and improvements can be proposed for them so that their industrial acceptance could increase.

The rest of this paper is organized as follows. Section II presents background and related work. Section III describes the SCAMPI-based assessment method which was used in the compliance analysis. Section IV presents the compliance analysis of OO-Method regarding TS. Finally, Section V summarizes some conclusions and proposes further work.

II. BACKGROUND

In this section, we provide a brief explanation of OO-Method, CMMI-DEV, and some works related to this paper.

A. OO-Method

OO-Method [8] is an object-oriented method for conceptual modeling and automatic code generation that is supported by the industrial tool *Olivanova* [12]. It provides a precise UML-like notation, which is used to specify a *Conceptual Schema* that describes a system at the problem space level. The development process suggested by OO-Method has two phases (Fig. 1): *Development of a conceptual schema* and *Generation of a software product*.

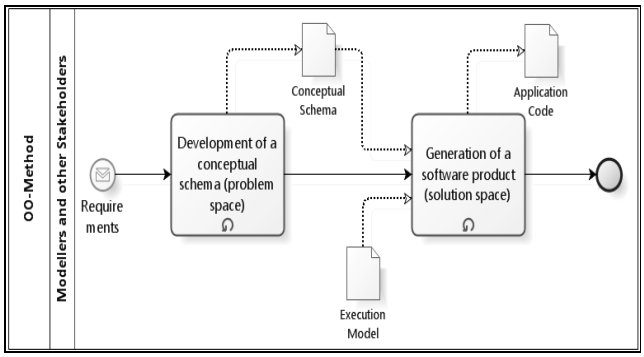

Figure 1. Phases and artifacts of the OO-Method MDD approach

The first phase consists of eliciting and representing the essential properties of the information system under study, thereby creating the corresponding conceptual schema. In the second phase, a precise execution model, conformed by a set of compilation patterns, indicates the correspondences between the conceptual schema and the pieces of code in a target implementation platform. Thus, the application code is automatically generated for an input conceptual schema.

B. CMMI-DEV

CMMI-DEV [10] is a guide to implement a continuous process improvement for developing products and services. For accomplishing this task, it provides two representations: ***Continuous***, which assesses the capability level of individual Process Areas (PAs) that are selected based on the organization's business goals; and ***Staged***, which assesses the maturity level of a whole development process.

The compliance analysis presented in this paper focuses on the continuous representation, since only one PA is analyzed. In this representation, capability levels have goals and practices (decomposition of goals) of two types: **1)** Specific Goals (SGs) and Specific Practices (SPs), which are applied only to a particular PA; and **2)** Generic Goals (GGs) and Generic Practices (GPs), which are applied equally to all PAs. From the assessment of practices and goals, it is possible to classify the capability level of a PA on a scale from 0 to 3 (unlike the previous versions of CMMI, the continuous representation no longer has capability levels 4 and 5). OO-Method will be assessed against level 1 because, according to CMMI-DEV, this level corresponds to the basis for improvement initiatives in a specific PA.

C. Related Work

Several authors have discussed the compliance of CMMI or CMM (the ancestor of CMMI) in relation to traditional or agile software development processes (e.g., [13][14][15], [16]). However, results obtained from these works are not completely useful to MDD approaches, which have characteristics that differentiate them from those other approaches [1][3][4][8]. For example, MDD approaches are mainly focused on: **1)** modeling rather than coding; **2)** implementing changes directly in the model rather than in the code; **3)** maintaining the model updated; **4)** synchronizing the model and the code; **5)** automatically verifying properties in the model; and **6)** automatically generating the complete code from models rather than using the model as a guide for manual code programming, or as a post-mortem code documentation.

In addition, at least one of the following problems can be found in the works related to those other software development approaches: **1)** non-use of the SCAMPI grades of satisfaction; **2)** lack of explicit/objective criteria for attributing grades; **3)** analysis based only on the activity descriptions, without requiring documental evidences or requiring only some of the evidences; **4)** analysis not in the same depth level as SCAMPI; **5)** lack of details about the rationale behind the analysis; or **6)** non-provision of solutions to fill in the gaps found in the analysis.

Although there are some specific works related to the compliance of MDD approaches with CMMI or CMM, they fail to deal with this issue properly. The works do not explain in detail how an approach complies with the quality model, where the approach should be adjusted for compliance, and whether/where the approach conflicts with the quality model requirements. The most relevant related works found are described as follows.

An engineering and management software process to support the achievement of CMM level 3 is proposed in [17]. The process uses MDA (a standard for MDD) [18] in the context of system families and CMM. However, an explicit mapping between the process and Key Process Areas (KPAs – a CMM concept that is equivalent to a PA of CMMI) is not presented, neither satisfaction grades are assigned.

The impacts in the software process and the main concerns to be dealt with when using MDD for the implementation of the CMM's KPAs are discussed in [19]. However, the discussion is presented in a high abstraction level, without providing any explicit compliance mapping neither the attribution of grades to KPAs.

The MDD Maturity Model, which establishes five capability levels towards the progressive adoption of MDD

 ISBN: 978-1-61208-165-6

within an organization, is proposed in [20]. The authors argue that this model is compliant and complementary to CMMI staged. However, they do not present evidences to justify this affirmation.

The compliance between a software development process (based on the formal specification language CSP-CASL) with the PAs of requirements management, product integration, requirements development, technical solution, verification, and validation of CMMI is analyzed in [21]. The analysis concluded that, in general, it is possible to conciliate the development based on formal specifications with the requirements of CMMI. Grades are assigned for the respective SPs and SGs of each analyzed PA. However, the grades are not based on the SCAMPI criteria and evidences are not presented to justify all of the assigned grades.

The Model-Driven (software) Development Process (MDDP) is presented in [22]. It covers the software development stages from business processes through system requirements, analysis, and design models into test scripts and code. The authors argue that it can be used to comply with every level of CMMI staged. But, a grading scheme and an explicit mapping identifying the documental evidences that give support to this compliance are not presented.

In [23], a case study on the use of MDD to support the implementation of processes for the following PAs of CMMI is described: requirements management, technical solution, product integration, verification, and validation. The study concluded that the use of MDD helps, but is not sufficient to satisfy all requirements of those PAs. However, evidences to justify this conclusion are not presented.

Unlike those works, we present a detailed compliance mapping which uses a SCAMPI-based grading method. The mapping was produced with the help of experts on the MDD approach and a CMMI consultant/member of assessment teams [1]. These facts reinforce the mapping validity.

III. A SCAMPI-BASED ASSESSMENT METHOD

SCAMPI [10] is a method to objectively assess the development process of an organization according to the requirements of respective PAs of CMMI. It deals with the consolidation of evidences (e.g., presentations, documents and interviews) related to the execution of the process in actual projects. The evidences are used, by an assessment team, to support the attribution of grades to practices, goals and, finally, to the evaluated PAs.

Although SCAMPI-based analyses are usually performed using artifacts from actual projects, we defined an assessment method based on existing publications on OO-Method [8][11][12][24] in order to obtain results independent from any organizational context, and draw conclusions without influences from the environment in which the approach is used. Therefore, these results and conclusions can be generalized to any organization that uses OO-Method or similar MDD approaches.

Assessment based on publications can be seen as a feature-based analysis performed as part of a major evaluation scenario [25]. For instance, an organization that follows or plans to follow CMMI might analyze the possibility of adopting a MDD approach as part of its development process; in this scenario, the organization can perform a feature-based analysis on each candidate MDD approach, and perform a preliminary selection (a subset of the candidate approaches) based on the analysis' results. Then, the selected approaches can be used on pilot projects to attest their effectiveness and to decide about the adoption.

The proposed SCAMPI-based assessment method uses the following types of evidences for the compliance analysis:

- Affirmations (*AFs*): statements described in the process that confirm or support implementation (or lack of implementation) of a practice as well as information obtained from experts in the approach.
- Artifacts (*ARs*): tangible evidences, mentioned in the process description, that are indicative of the work being performed and represent either the primary outputs of a model practice or a consequence of implementing a model practice.

The assessment is performed on a bottom-up way, from the practices up to the goals. Hence, for characterizing the level of implementation of a Specific Practice (SP) or Generic Practice (GP), the following grades are used:

- Fully Implemented (FI): *ARs* are present and judged to be adequate for demonstrating the practice implementation. No weaknesses are found.
- Largely Implemented (LI): *ARs* are present and judged to be adequate for demonstrating the practice implementation. However, one or more weaknesses are found.
- Partially Implemented (PI): some or all data required is absent or judged to be inadequate, some data provided (if exist) suggest that aspects of the practice are implemented, and one or more weaknesses are found; or the data supplied to the assessment team present conflicts, i.e., certain data indicate that the practice is implemented and other data indicate the practice is not implemented, and also, one or more weaknesses are noted.
- Not Implemented (NI): some or all data required is absent or judged to be inadequate, data supplied (if exist) do not support the conclusion that the practice is implemented, and one or more weaknesses are noted.

Based on the grades defined for a practice, each Specific Goal (SG) or Generic Goal (GG) is graded as:

- Satisfied: if and only if all corresponding practices are graded as either LI or FI, and the aggregation of weaknesses associated with the goal does not have a significant impact on the goal achievement; or,
- Unsatisfied: if at least one of the corresponding practices has a grade different from LI or FI.

Based on the grades defined for the goals (SGs and GGs), the capability level of a PA is defined. For instance, the capability level 1 has to satisfy the associated GG (hereafter called GG 1). Furthermore, GG 1 is "Satisfied" if all the SGs associated to the PA are graded as "Satisfied".

 ISBN: 978-1-61208-165-6

For capability levels higher than 1, a PA must satisfy the GG associated to the current level as well as all the GGs associated to the lower levels. The evaluation of the current GG is performed by applying the grading method defined previously to all the GPs associated to the GG (for instance, a process has capability level 2 for a specific PA if it satisfies all the GPs associated to the GG of capability level 2 and also satisfies GG 1). The application of this SCAMPI-based assessment method to OO-Method is shown in next section.

IV. COMPLIANCE ANALYSIS OF OO-METHOD WITH THE TECHNICAL SOLUTION PROCESS AREA

The purpose of TS is providing guidance for design, development and implementation of the given product requirements [9]. It focuses on evaluating and selecting a solution, developing a detailed design of the solution, and implementing the design as a product or product component.

The compliance analysis presented in this section uses an instance of the assessment method described in Section III. Based on publications about OO-Method, the CMMI expert carried out the analysis playing the role of an assessor. Then, experts on the MDD approach reviewed the analysis to identify possible flaws or misinterpretations. After that, discussions on the results were carried out by the experts (on CMMI and on the MDD approach) to validate the analysis.

The following subsections detail the results of the compliance analysis of OO-Method regarding the capability level 1 of TS process area. For each SG, the purpose of each corresponding SP is presented, the practice is mapped to *AFs* and *ARs*, and the SP is graded. After grading all the SPs of a SG, each SG is graded. Finally, a summary of the results is presented, where the whole PA is graded. Improvement suggestions are also discussed.

A. SG 1 Select Product Component Solutions

This specific goal includes SP1.1 and SP1.2.

SP 1.1 Develop Alternative Solutions and Selection Criteria

This SP identifies and analyzes alternative solutions to enable the selection of a balanced solution in terms of cost, schedule, performance, and risk. Selection criteria typically address costs (e.g., time, people, money), benefits (e.g., product performance, capability, effectiveness), and risks (e.g., technical, cost, schedule).

AFs: The application's architecture is defined based on a three-layer architectural pattern and restrictions on the selected technological platform, implementation language(s), and persistence service(s). Then, the conceptual model compilation is parameterized by the chosen architecture and the application's source code can be automatically generated [8].

ARs: Conceptual Schema, Execution Model, Application Code.

Grade: PI

OO-Method does not explicitly analyze alternative solutions prior to code generation (the architecture is usually defined by the application's developers jointly with the clients); neither defines criteria for the architecture selection. However, if the source code generated is not adjusted to the quality requirements of a particular application, it can be regenerated for alternative platforms [8].

SP 1.2 Select Product Component Solutions

This SP selects the product component solutions based on selection criteria. Lower level requirements are generated from the selected architecture and used to develop product component designs. Interfaces among product components are described. The description of the solutions and the rationale for selection are documented.

AFs: Components identified during the phase "*Development of a conceptual schema*" are allocated to the architecture layers [8]. A documentation manager [26], which is part of the suite of tools to support OO-Method, automatically generates documentation from the conceptual schema, describing each component and its interface with other components.

ARs: Conceptual Schema, Execution Model, Application Code, Generated documentation, Documentation manager tool.

Grade: LI

OO-Method does not have an explicit artifact for documenting the selection decisions that are related to product component solutions, neither their rationale.

Conclusion

SP1.1 is graded as PI and SP1.2 is graded as LI. Therefore, SG 1 is graded as "Unsatisfied".

B. SG 2 Develop the Design

This specific goal includes SP2.1, SP2.2, SP2.3, and SP2.4.

SP 2.1 Design the Product or Product Components

This SP designs the product or its components in two phases, which can overlap in execution: preliminary (abstract) and detailed (concrete) design.

AFs: During the phase "*Development of a conceptual schema*", an abstract architecture is defined. A concrete architecture is defined during the phase "*Generation of a software product*", according to an execution model driven by restrictions on the selected technological platform, implementation language(s), and persistence service(s) [8].

ARs: Conceptual Schema, Execution Model, Application Code.

Grade: FI

No weak points have been identified.

SP 2.2 Establish a Technical Data Package

This SP records the design in a technical data package (a collection of items providing the developer a description of the product or product components) created during preliminary design.

AFs: A documentation manager is responsible for documenting the architecture definition from the conceptual schema created, and a repository

 ISBN: 978-1-61208-165-6

manager is responsible for creating and administrating a model library (including access control and management of model versions) [26].

ARs: Generated documentation, Data repository, Documentation manager tool, Repository manager tool, Conceptual Schema, Execution Model, Application Code.

Grade: FI

No weak points have been identified.

SP 2.3 Design Interfaces Using Criteria

This SP designs product component interfaces using established criteria (e.g., critical parameters that should be defined, or at least investigated, to ascertain their applicability).

AFs: OO-Method defines criteria for validating the Conceptual Schema in terms of correctness and completeness. The specification samples presented in part II of [8] illustrate the use of these criteria. Additional criteria for evaluating consistency, correctness and completeness are defined in [24].

ARs: Conceptual Schema, Validation criteria, Execution Model, Application Code.

Grade: FI

No weak points have been identified.

SP 2.4 Perform "Make, Buy, or Reuse" Analysis

This SP evaluates whether the product components should be developed, purchased, or reused based on established criteria.

AFs: Not identified.

ARs: Not identified.

Grade: NI

OO-Method does explicitly mention this analysis in its process; neither establishes criteria for performing it. However, the approach makes it possible to integrate developed components with pre-existing components and/or systems (legacy code).

Conclusion

SPs 2.1, 2.2 and 2.3 are graded as FI, and SP 2.4 is graded as NI. Therefore, SG 2 is graded as "Unsatisfied".

C. SG 3 Implement the Product Design

This specific goal includes SP3.1, SP3.2, and SP3.3.

SP 3.1 Implement the Design

This SP implements the design of the product components and includes the allocation, refinement, and verification of each product component.

AFs: Once architecture is chosen, its code can be automatically generated. Prior to the source code generation, it is possible to have a conceptual model validation [8].

ARs: Conceptual Schema, Validation criteria, Execution Model, Application Code.

Grade: FI

No weak points have been identified.

SP 3.2 Develop Product Support Documentation

This SP develops and maintains documentation that will be used to install, operate, and maintain the product.

AFs: The documentation manager [26] automatically generates support and end-user documentation from the conceptual schema created.

ARs: Generated documentation, Documentation manager tool, Conceptual Schema, Application Code.

Grade: FI

No weak points have been identified.

Conclusion

SP 3.1 is graded as FI and SP 3.2 is graded as FI. Therefore, SG 3 is graded as "Satisfied".

D. Summary of Assessment and Improvement Suggestions

As Table I summarizes, the OO-Method development process has capability level 0 with regard to TS.

TABLE I. OVERALL RESULTS OF OO-METHOD ASSESSMENT

Goals and practices of TS	Grades
SG 1 Select Product Component Solutions	Unsatisfied.
SG 2 Develop the Design	Unsatisfied.
SG 3 Implement the Product Design	Satisfied

In spite of this negative result, several convergence points have been pointed out, and most of the weak points found are easy to solve with the improvement suggestions described in Table II. In general, the improvements are simple adjustments in the development process, mainly related to explicit documentation of evidences.

TABLE II. IMPROVEMENT SUGGESTIONS FOR OO-METHOD

Improvements	Affected SPs
Extension of the development process to include an explicit analysis of alternative solutions prior to the code generation, as well as the explicit creation of a document defining criteria for the architecture selection.	SP 1.1
Explicit documentation of selection decisions (related to product component solutions) and their rationale.	SP 1.2
Explicit definition of criteria to perform "make, buy or reuse" analysis and the creation of an activity to explicitly perform this analysis prior to the code generation.	SP 2.4

Thus, by tailoring OO-Method with these improvements, it is possible to turn the grade of all SGs of TS into "Satisfied". As a result, the development process of OO-Method can reach the capability level 1 for this PA. However, in order to confirm the effectiveness of the changes, the improvements should be implemented in a new version of OO-Method and the modified approach should be used in actual projects.

V. CONCLUSIONS AND FUTURE WORK

Even though this work has presented an analysis for CMMI and OO-Method, its purpose is to emphasize the need of analyzing the compliance of MDD approaches in relation to any software process quality model and to show how it can be addressed. Hence, the analysis can be adapted to other quality models [6][7] and to other MDD approaches [27].

 ISBN: 978-1-61208-165-6

In this paper, the compliance of OO-Method in relation to TS process area was analyzed. As a result, it was detected that this MDD approach does not sufficiently implement certain SPs. Hence, improvements were proposed aiming to fully implement the SPs for which weaknesses were found.

Some of the problems found in the analysis of OO-Method are also common to other approaches [27]. For instance, the lack of "make, buy, or reuse" analysis is a problem found in most of the approaches. Moreover, some of the features that were satisfied for OO-Method (e.g., the full code and document generations) are not presented in all of the other approaches [27].

This work is a starting point for several future works. We plan to perform a SCAMPI-based analysis of other MDD approaches for assessment and comparison; an assessment of the OO-Method development process applied to real projects must be addressed and an evaluation of the proposed improvements must be performed; the compliance with other PAs and capability levels of CMMI-DEV will be analyzed against OO-Method; and a systematic literature review [28] will be conducted to verify the existence of other works related to the compliance of MDD and software process quality models. All of these works are part of a research agenda towards the development of a complete MDD-based software process compliant with CMMI.

ACKNOWLEDGMENT

This work has been developed with the support of the Brazilian Research Agency CAPES, under the grant #BEX3229/10-6, and the Spanish Government, under the projects ORCA (PROMETEO/2009/15) and PROS-REQ (TIN2010-19130-C02-02).

REFERENCES

[1] A. Vasconcelos, "Curriculum Vitae", in Portuguese, http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4787134E7, Last access 2011/07/18

[2] B. Selic, "The Pragmatics of Model-Driven Development," IEEE Software, vol. 20(5), 2003, pp. 19-25

[3] D. Schmidt, "Model-Driven Engineering," IEEE Computer, vol. 39(2), 2006, pp. 25-31

[4] M. Völter, "Model-Driven Software Development – Technology, Engineering, Management," Wiley, 2007

[5] P. Mohagheghi and V. Dehlen, "Where is the Proof? – A Review of Experiences from Applying MDE in Industry," LNCS, Springer, 2008, vol. 5095, pp. 432-443

[6] F. J. Pino, F. García, and M. Piattini, "Software process improvement in small and medium software enterprises: a systematic review," Software Quality Journal, vol. 16(2), 2008, pp. 237-261

[7] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. Cheng, R. Permadi, and R. Feldt, "Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review," IEEE-TSE, 2011

[8] O. Pastor and J. C. Molina, Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, 1st edn. Springer, New York, 2007

[9] SEI, "CMMI for Development, Version 1.3," CMU/SEI-2010-TR-033, http://www.sei.cmu.edu, 2010, Last access 2011/07/18

[10] SEI, "Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document", CMU/SEI-2011-HB-001, 2011, http://www.sei.cmu.edu, 2010, Last access 2011/07/18

[11] PROS, Model Driven Development and Automatic Code Generation, http://www.pros.upv.es/index.php/en/lineas/69-lineaddm, Last access 2011/07/18

[12] O. Pastor, J. C. Molina, and E. Iborra, "Automated production of fully functional applications with OlivaNova Model Execution", ERCIM News, nº 57, 2004

[13] L. V. Manzoni and R. T. Price, "Identifying Extensions Required by RUP to Comply with CMM Levels 2 and 3," IEEE TSE, vol. 29(2), 2003, pp. 181-192

[14] C. Santana, C. Gusmão, A. Rouiller, and A. Vasconcelos, "Achieving Software Quality Certifications through Agile Software Development," Internat. Journal of Advanced Manufacturing Systems, vol. 11(1), 2008, pp. 1-6

[15] J. Smith, "Reaching CMM Level 2 and 3 with the Rational Unified Process – White Paper," http://www.wthreex.com/rup/portugues/papers/pdf/rupcmm.pdf, 2000, Last access 2011/07/18

[16] J. Diaz, J. Garbajosa, and J. A. Calvo-Manzano, "Mapping CMMI Level 2 to Scrum Practices: An Experience Report," Proc. EuroSPI 2009, CCIS, Springer, 2009, vol. 42, pp. 93-104

[17] ESI, "Model-driven Architecture inSTrumentation, Enhancement and Refinement," IST-2001-34600, MASTER-2003-D3.2-V1.0-PUBLIC, 2003

[18] OMG, "Model Driven Architecture (MDA) Guide Version 1.0.1," http://www.omg.org/mda/, 2003, Last access 2011/07/18

[19] R. Steinhau, et al., "Guidelines for the Application of MDA and the Technologies covered by it," Deliverable 3.2, MODA-TEL Consortium, IST-2001-37785, Interactive Objects Software GmbH, 2003

[20] E. Rios, T. Bozheva, A. Bediaga, and N. Guilloreau, "MDD Maturity Model: A Roadmap for Introducing Model-Driven Development. Model Driven Architecture – Foundations and Applications," LNCS, Springer, 2006, vol. 4066, pp. 78-89

[21] S. Mishra and B. Schlingloff, "Compliance of CMMI Process Area with Specification Based Development," Proc. VI Internat. Conf. on Softw. Engineering Research, Management and Applications, IEEE Computer Society, 2008, pp. 77-84

[22] Crag Systems, The Model-Driven Development Process, http://www.cragsystems.co.uk/development_process, 2008, Last access 2011/07/18

[23] S. Fricker,"Introducing Model-Driven Development for CMMI Engineering Process Areas," SEPG 2006, http://www.secc.org.eg/sepg%202006/ingredients/Indexes/authorindex.html#f, 2006, Last access 2011/07/18

[24] B. Marín, G. Giachetti O. Pastor, and A. Abran, "A Quality Model for Conceptual Models of MDD Environments" Advances in Software Engineering, (Special Issue: New Generation of Software Metrics), 2010

[25] B. Kitchenham, "DESMET: A Method for Evaluating Software Engineering Methods and Tools," TR96-09, Department of Computer Science, University of Keele, 1996

[26] CARE-Technologies Web Page, http://www.care-t.com, 2011/07/18

[27] J. Estefan, "Survey of Model-Based Systems Engineering Methodologies – Rev. B," Technical Report, INCOSE MBSE Initiative Focus Group, 2008

[28] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3," Keele University and Durham University Joint Technical Report EBSE-2007-01, 2007

 ISBN: 978-1-61208-165-6