Certifying CMMI-DEV in the Context of Model-Driven Software Engineering

CLAUDIA PONS AND VIVIANA ESTERKIN

Model-driven software engineering (MDE) is being positioned as an alternative to conventional methods of software production. Given that MDE is an emerging paradigm, standards for measuring its quality have not yet been established. This article analyzes MDE good practices and how they relate to CMMI-DEV 1.3 Level 2. MDE best practices were assessed to determine whether they support each CMMI Level 2 specific practice in seven of the 22 process areas: configuration management, supplier agreement management, requirement management, process and product quality assurance, measurement and analysis, project monitoring and control, and project planning. An expert panel of five software engineering professionals offered consulting services to provide an initial evaluation of the results. For each process area, the percentage of practices supported by MDE was determined and recommendations to enhance MDE support were identified. Although further research is needed, this suggests that an organization that uses MDE can certify at CMMI-DEV 1.3 Level 2.

KEY WORDS

Capability Maturity Model, CMMI-DEV 1.3, model-driven engineering (MDE), quality assessment

INTRODUCTION

Model-driven software engineering (MDE) (Brambilla, Cabot, and Wimmer 2012; Stahl and Voelter 2006) has been proposed as an alternative to more conventional methods of software production. It presents a new way of understanding development and maintenance of software systems by using models as the primary artifacts in the development process. In MDE, the models are used to direct tasks related to comprehension, design, construction, tests, deployment, operation, management, maintenance, and modification of systems. Several examples of the successful introduction of MDE have been provided by Di Ruscio, Paige, and Pierantonio (2014) and Object Management Group (2015), who report on the existing use of tools that make this approach real in industry today.

The Capability Maturity Model (CMMI) (SEI 2010) is defined as the integration of a set of models for evaluation, and improvement of the processes for the development, maintenance, and operation of systems. It provides guidelines for applying a group of best practices to these processes. It is managed by the Software Engineering Institute (SEI) from Carnegie Mellon University, and is considered the de-facto standard to evaluate quality of software system development practices. It consists of five maturity levels that indicate the sophistication reached by the organization in its software development processes, from Level 1 (initial) to Level 5 (optimizing). Additionally, the maturity levels can be used to assess organizational improvement relative to one of 22 process areas.

The aim of the authors' study is to identify the support that MDE gives to CMMI-DEV Level 2. To this end, the seven process areas associated with Level 2 are analyzed and described in terms of specific practices, that when implemented, are projected to satisfy their goals. The seven areas are:

1. Configuration Management (CM)
2. Supplier Agreement Management (SAM)
3. Requirements Management (REQM)
4. Process and Product Quality Assurance (PPQA)
5. Measurement and Analysis (MA)
6. Project Monitoring and Control (PMC)
7. Project Planning (PP)

Given that MDE is an emerging paradigm, standards for measuring the quality of its applications have not been established yet. This article provides a contribution in this regard, analyzing MDE good practices in relation to a well-established quality evaluation model. In particular, this article summarizes 50 "good practices" of MDE identified by the literature and relates them to the specific practices in CMMI-DEV 1.3 Level 2, to help practitioners understand how the two approaches compare to each other.

This article is organized as follows: a) MDE "good practices" proposed by the literature are outlined; b) MDE content is analyzed to determine whether MDE provides support to the specific practices defined by CMMI-DEV 1.3 Level 2 in the process areas that correspond to that level; c) preliminary validation of results by expert software engineers is performed; and d) a description of the process areas supported by MDE and the degree of support is presented. The evaluation ends with a discussion of proposals that would increase MDE support for CMMI-DEV 1.3 Level 2.

BACKGROUND

Capability Maturity Models

The CMMI has two representations that allow the organization to achieve different improvement goals: the staged representation and the continuous representation. The presentation and organization of the information differs in both representations; however, the content is the same. In this study, CMMI for Development (CMMI-DEV) version 1.3 is used.

CMMI-DEV consists of five maturity levels that indicate the sophistication reached by the organization in its software development processes. The maturity levels describe an evolutionary path that an organization that wants to improve its processes to develop products or services can employ. The five maturity levels are designated as follows:

- **Level 1: Initial.** At maturity level 1, the organization does not provide a stable environment to support processes. Although the appropriate engineering techniques may be used, efforts can be weakened by a lack of formal plans. The results of a project can be unpredictable.
- **Level 2: Managed.** At maturity level 2, organizations provide institutionalized practices of project management, with basic metrics and reasonable follow-up of quality performance.
- **Level 3: Defined.** At maturity level 3, in addition to good project management, the organization provides appropriate coordination procedures among groups, staff training, more detailed engineering techniques, and more advanced metrics for processes.
- **Level 4: Quantitatively managed.** At maturity level 4, the organization provides a set of significant quality and productivity metrics and uses its quality system systematically for decision making and risk management.
- **Level 5: Optimizing.** At maturity level 5, the whole organization is devoted to continual improvement of the processes. Metrics are intensively used and the innovation process is actively managed.

A process area is defined as a set of related practices that, when implemented collectively, satisfies a set of goals considered important for making improvements in that area.

Model-Driven Software Engineering

MDE is an approach to software development that uses models as primary artifacts, from which code, documentation, and tests are derived (Brambilla, Cabot, and Wimmer 2012). MDE proposes the solution of current software development problems by using a framework ensuring portability, interoperability, platform independence, and productivity. Moreover, model-driven architecture

(MDA) (Kleppe, Warner, and Bast 2003) was created to give support to model-driven development. MDA is an architecture that provides a set of guidelines to structure specifications expressed as models. Using the MDE/MDA methodology, the system's functionality will be defined, in the first instance, as a platform-independent model (or PIM), through a specific language for the domain under study. The PIM model can be translated to one or more platform-specific models (PSMs) for the corresponding implementation. Translation from PIM to PSMs is normally conducted using automated tools for model transformation.

MDE can have a deep impact on the software construction process. Organizations and projects frequently depend on experts who make decisions related to the system. MDE enables capturing their experience within the models and transformations, thus allowing other members of the team to take advantage of expert knowledge without demanding their physical presence. Moreover, this tacit and explicit knowledge can be maintained more easily, even when experts leave the organization. In addition, development and testing costs can be reduced significantly when automating a large part of the work related to code (and other artifacts) in this manner. By means of automation, MDE favors the consistent generation of artifacts, and reduces the presence of errors.

GOOD PRACTICES IN MDE

The selection of good practices for MDE was conducted by performing an extensive literature review, from which three candidate papers that best aggregated such practices were selected: Swithinbank et al. (2005), Pons, Giardini, and Perez (2010), and Rios et al. (2006). Each practice selected is identified with the acronym GP (good practice) and a number.

Practices Extracted From Swithinbank et al. (2005)

- **GP1: Identify common patterns and standards.** The solution architect identifies the patterns repeated in business applications. These patterns arise many times due to the consistent use of an architectural style or due to requirements of the execution platforms.
- **GP2: Identify reusable MDE assets.** In this task the solution architect compares the common patterns identified in task GP1 with the existing MDE assets, making the necessary architectural adjustments to exploit what is already available. The assets can come from previous MDE projects or standard elements.
- **GP3: Define the design model.** The solution architect chooses the appropriate type of Unified Modeling Language (UML) model for the application developers. This model will be used when the specific details of the components that are being built are defined. It also creates an initial list of stereotypes for the UML profile.
- **GP4: Identify the platform-independent model.** This model can be reformed by the solution architect or by an experienced developer who understands the execution environment.
- **GP5: Produce sample devices for key scenarios.** An application programmer manually programs the devices that will act as detailed plans for templates and transformations.
- **GP6: Define the MDE tool chain.** The task identifies the MDE tools needed for the project development. Once the task is completed, it is possible to create a detailed plan of the effort demanded to build the MDE tool chain.
- **GP7: Validate the tool chain.** This task is performed by the solution architect, who is responsible for the MDE project.
- **GP8: Requirements for the validation of the tool chain.** A business application developer should not modify an MDE artifact already generated; tools must be totally integrated with the configuration management system
- **GP9: Automatic generation of devices.** It will be possible to regenerate all the artifacts of the business application automatically from a file generated to that end. Thus, if it is necessary to partly enlarge a transformation during the construction of a business application, everything can be regenerated automatically.
- **GP10: Follow-up and control.** Once the project plan has been built, follow-up and control of an MDE project does not differ from that of other software development projects.
- **GP11: Successful reutilization.** Success of an MDE project depends on the success of the

reutilization of artifacts. This includes the identification and recovery of an artifact to be reused; certainty that the appropriate artifact is being recovered for the corresponding execution version; checking the integrity of the artifact; and verifying if it is the appropriate version.

- **GP12: The follow-up.** Follow-up of an MDE project is similar to that of any other software project. However, there are some additional advantages that MDE adds that are derived from its automation.
- **GP13: Life cycle of a project.** The framework covers the creation, testing, and development of models, patterns, and transformations that will generate the solutions.
- **GP14: Versions.** There must be a mechanism for the development and substitution of new versions that can co-exist and ensure they are available for the appropriate customer.
- **GP15: Versioning level.** The versioning level (by file, class, service, development unit, and others) to be applied must be determined. Transformations, patterns, profiles, and other reusable devices are versioned.
- **GP16: Service certification of the model or artifact.** It is recommended to have a mechanism to certify that artifacts and models meet the standards and that the integrity of the system is maintained.
- **GP17: Model depuration.** The code generated must not be depurated. Models and transformations must be depurated instead for two reasons: a) it is extremely difficult to return from the code to the problem underlying in the model; b) it is crucial that all the changes are conducted in the models or transformations *and not in the generated artifacts*. This practice ensures consistency of the models and the generated solution.
- **GP18: Validation and testing.** Solution artifacts must be validated against the requirements of the solution and the business logic of the services. MDE testing includes two phases: a) testing the model's framework; and b) testing the solution artifacts generated.
- **GP19: Valid information.** Models and transformations in an MDE development must be built with accurate and valid information.

Practices Extracted From Pons, Giardini, and Pérez (2010)

- **GP20: Experts.** The MDE platform must be developed by the most experienced professionals: domain experts, language developers, modelers or engineers, transformation developers, and/or code generation developers.
- **GP21: Iterations.** It is recommended to separate the development into several iterations.
- **GP22: Guidelines.** It is recommended to take into account the following guidelines during project development: a) explicitly invest in support tools; b) employ the best qualified people to develop MDE tools with the purpose of capturing and automating their experience; c) consider that in addition to the code, the project will generate documents, configurations, reports, and test cases; d) ensure that the development process supports testing environments in addition to production environments; e) define configuration management strategies for MDE tools; f) assign a period of time for the team training on the use of MDE tools; g) assign a period of time to consider whether the MDE tools will be reusable in future projects.
- **GP23: Metrics.** On completing the MDE project, it is useful to generate metrics for assessing the cost of tool development and the productivity of the application developers when using the tools compared with the effort that would be needed to develop the whole code manually.
- **GP24: Tools.** Identify, develop, and install the MDE tools required, before the business application developers need them.
- **GP25: Management.** MDE artifacts, their related descriptions, and their repositories must be actively managed.

Practices Extracted From Ríos et al. (2006)

In Ríos et al. (2006) the authors define a maturity model for MDE introduction into an organization. This model consists of five capability levels. Maturity level 1 corresponds to situations where modeling practices are sporadically used or not used. For maturity level 2, named basic MDE, the following practices are defined:

- **GP26: Modeling techniques.** Identify modeling techniques.
- **GP27: Technical model.** Define the technical model.
- **GP28: Code generation.** Generate a code from a technical model.
- **GP29: Documentation.** Generate documentation from the technical model.
- **GP30: Complete code.** Complete code to comply with all requirements.
- **GP31: Selection of tools.** Decide upon appropriate modeling tools.

According to these authors, in maturity level 3 (named initial MDE) the organization starts developing systems in a more model-driven manner. Besides aligning the code and the models, it develops business models that address the business logic of the system separately from the technical models. Business models are then manually converted to technical models, but technical models are represented by means of a tool and can be converted to code automatically. For maturity level 3, the following practices are defined:

- **GP32: Model.** Define business model.
- **GP33: Transformations.** Define transformations from technical model to code.
- **GP34: Separation in the generated code.** Separate generated from nongenerated code.
- **GP35: Checking.** Check models.
- **GP36: Workflow.** Define MDE-project workflow.
- **GP37: Coverage.** Decide upon coverage of modeling activities.
- **GP38: Repositories.** Establish and maintain repositories for models and transformations.
- **GP39: Measures.** Define, collect, and analyze measures with respect to modeling activities.

In maturity level 4 (named integrated MDE) the organization begins integrating its models. Business models are derived from the domain models and are developed by means of a tool. They are automatically transformed to technical models, and these technical models become code. Domain, business, and technical concepts are separated. For maturity level 4, the following good practices are defined:

- **GP40: Metamodel.** Define architecture centric metamodel.
- **GP41: Domain model.** Define the domain model.
- **GP42: Transformations.** Define the transformations from business model to technical model.
- **GP43: Simulation.** Simulate the models.
- **GP44: Separation.** Separate the technical models of the product from the system family infrastructure.
- **GP45: Infrastructure management.** Manage common infrastructure development.

In maturity level 5 (final MDE) transformations between models are made automatically, and the models are fully integrated with code. For maturity level 5, the following good practices are defined:

- **GP46: DSLs.** Define domain-specific languages.
- **GP47: Improvement and validation of the metamodel.** Continuously improve and validate metamodels.
- **GP48: Transformations.** Define transformations from domain model to business model.
- **GP49: V&V.** Model-based validation and verification.
- **GP50: Strategic elements.** Establish and maintain strategic MDE elements.

ANALYSIS OF MDE GOOD PRACTICES IN THE CONTEXT OF CMMI-DEV

This section analyzes that MDE practices support each process area. To this end, the authors look for activities, artifacts, workflows, procedures, or people implementing the specific practices of each area in MDE. To identify

each specific practice, the SP acronym will be used, followed by a number ("x.y"). The "x" is the number of the specific goal to which the specific practice corresponds. The "y" is the sequence number of the specific practice within that goal. This terminology is used throughout the study to refer to the specific practices in CMMI-DEV 1.3. Table 1 shows the specific practices of the configuration management process area sorted by specific goal.

Not all good MDE practices selected have been used to evaluate CMMI-DEV 1.3 Level 2; however, the purpose of the complete list is to start the analysis for the remaining CMMI-DEV 1.3 levels that are beyond the scope of the present study. To facilitate the understanding of the whole analysis that will be introduced in the following section, the detailed examination of two specific practices is described here as an example.

- Example 1: In the configuration management process area, SP1.1 states "Identify configuration items." MDE provides support to this practice through the following practices: practices GP1/GP6 identify MDE artifacts (or configuration items) that must be generated; practice GP24 indicates that MDE artifacts must be identified before the developers of business application need them; practices GP26/GP30, GP40/GP43, GP46, and GP48 indicate MDE artifacts/configuration items

TABLE 1 Specific practices by goal

Configuration Management (CM) Process Area
SG1 Establish Baselines
SP1.1 Identify configuration Items
SP1.2 Establish a CM System
SP1.3 Create or release baselines
SG2 Track and Control Changes
SP2.1 Track change requests
SP2.2 Control configuration items
SG3 Establish Integrity
SP3.1 Establish CM records
SP3.2 Perform configuration audits
Requirements Management (REQM) Process Area
SG1 Manage Requirements
SP1.1 Understand requirements
SP1.2 Obtain commitment to requirements
SP1.3 Manage requirements changes
SP1.4 Maintain bidirectional traceability of requirements
SP1.5 Ensure alignment between project work and requirements
Process and Product Quality Assurance (PPQA) Process Area
SG1 Objectively Evaluate Processes and Work Products
SP1.1 Objectively evaluate processes
SP1.2 Objectively evaluate work products
SG2 Provide Objective Insight
SP2.1 Establish records
SP2.2 Communicate and resolve noncompliance issues

Measurement and Analysis (MA) Process Area
SG1 Align Measurement and Analysis Activities
SP1.1 Establish measurement objectives
SP1.2 Specify measures
SP1.3 Specify data collection and storage procedures
SP1.4 Specify analysis procedures
SG2 Provide Measurement Results
SP2.1 Obtain measurement data
SP2.2 Analyze measurement data
SP2.3 Store data and results
SP2.4 Communicate results
Project Monitoring and Control (PMC) Process Area
SG1 Monitor the Project Against the Plan
SP1.1 Monitor project planning parameters
SP1.2 Monitor commitments
SP1.3 Monitor project risks
SP1.4 Monitor data management
SP1.5 Monitor stakeholder involvement
SP1.6 Conduct progress reviews
SP1.7 Conduct milestone reviews
SG2 Manage Corrective Action to Closure
SP2.1 Analyze issues
SP2.2 Take corrective action
SP2.3 Manage corrective actions

Project Planning (PP) Process Area
SG1 Establish Estimates
SP1.1 Estimate the scope of the project
SP1.2 Establish estimates of work product and task attributes
SP1.3 Define project life-cycle phases
SP1.4 Estimate effort and cost
SG2 Develop a Project Plan
SP2.1 Establish the budget and schedule
SP2.2 Identify project risks
SP2.3 Plan data management
SP2.4 Plan the project's resources
SP2.5 Plan needed knowledge and skills
SP2.6 Plan stakeholder involvement
SP2.7 Establish the project plan
SG3 Obtain Commitment to the Plan
SP3.1 Review plans that affect the project
SP3.2 Reconcile work and resource levels
SP3.3 Obtain plan commitment
Supplier Agreement Management (SAM) Process Area
SG1 Establish Supplier Agreements
SP1.1 Determine acquisition type
SP1.2 Select suppliers
SP1.3 Establish supplier agreements
SG2 Satisfy Supplier Agreements
SP2.1 Execute the supplier agreement
SP2.2 Accept the acquired product
SP2.3 Ensure transition of products

that must be generated during development. In summary, there are MDE practices that, when accomplished, satisfy the goal of the CMMI practices mentioned.

- Example 2: In the requirements management process area, SP1.5 defines "Ensure alignment between work products and requirements." In this case, MDE support is based on the practice GP5, which indicates that simple artifacts must be produced for key scenarios, and on the practice GP6, which states the need of validating the tool chain to ensure the alignment of work products with the requirements. Moreover, the application of GP49 ensures that traceability and alignment will be maintained between work products and requirements. Therefore, there are MDE practices that, when accomplished, satisfy the goal of the CMMI practices mentioned.

The results obtained for each of the process areas CMMI-DEV 1.3 Level 2 are as follows.

Configuration Management (CM) Process Area

According to CMMI-DEV, the purpose of this process area is to establish and maintain the integrity of work products using configuration identification, configuration control, and configuration status and configuration audits. After analyzing the specific practices here, the authors conclude that there are several MDE good practices that support this process area. Table 2 shows the GPs that give support to each SP in the area. The authors conclude that seven CMMI-DEV 1.3 specific practices out of seven are supported by MDE.

Requirements Management (REQM) Process Area

According to CMMI, the purpose of this process area is to manage requirements of the project's products and components and to ensure alignment between those requirements and the project's plan and work products. In this case, the MDE support is complete, given that handling requirements in an MDE project means defining the characteristics and management of the main MDE artifacts (that is, the models), and the procedures for carrying out the modeling activity are described by all the authors who were taken as reference. Table 3 shows GPs that give support to each SP in the area. This process area has five specific practices. All of them are supported by MDE.

Process and Product Quality Assurance (PPQA) Process Area

The purpose of this area is to provide staff and management with objective insight into processes and associated work products. A high MDE support has been verified to this process area, as can be observed in data displayed in Table 4 on the next page. This process area has four specific practices, three of which are supported by MDE.

Measurement and Analysis (MA) Process Area

The purpose of measurement and analysis is to develop and sustain a measurement capability used to support

TABLE 2 MDE support for configuration management (CM) process area

SP	Definition of the SP	GPs that support it
1.1	Identify the configuration items	GP1-6, GP24, GP26-30, GP40-43, GP46, GP48
1.2	Establish a configuration management system	GP22, GP25, GP8, GP9, GP11, GP14, GP38, GP50
1.3	Create baselines	GP1-6, GP27-33, GP36, GP40-42
2.1	Track changes requirements	GP9
2.2	Control the configuration items	GP9, GP17, GP18
3.1	Establish records for configuration management	GP8, GP9, GP50
3.2	Conduct configuration audits	GP8

TABLE 3 MDE support for requirements management (REQM) process area

SP	Definition of the SP	GPs that support it
1.1	Understand the requirements	GP3, GP5, GP13, GP20
1.2	Obtain commitment to requirements	GP6, GP13, GP22
1.3	Manage requirement changes	GP14 y GP15
1.4	Maintain bidirectional traceability of requirements	GP11, GP15, GP16, GP18, GP25, GP49, GP50
1.5	Ensure alignment between work products and requirements	GP5, GP6, GP49

TABLE 4 MDE support for process and product quality assurance (PPQA) process

SP	Definition of the SP	GPs that support it
1.1	Objectively evaluate processes	MDE GP6, GP8, GP13, GP16, GP22, GP35, GP50
1.2	Objectively evaluate work products	GP5, GP11, GP16, GP18, GP22, GP25
2.1	Establish records	GP11, GP16, GP50
2.2	Communicate and resolve noncompliance issues	Not supported

TABLE 5 MDE support for each specific practice of the MA process area

SP	Definition of the SP	GPs that support it
1.1	Establish measurement objectives	GP23
1.2	Specify measures	GP23
1.3	Specify data collection and storage procedures	GP9, GP11, GP25, GP38

TABLE 6 MDE support to each process area of the CMMI-DEV Level 2

Process Area	Total number of SPs	Number of SPs supported by MDE	% supported by MDE
Configuration management (CM)	7	7	100%
Requirements management (REQM)	5	5	100%
Process and product quality assurance (PPQA)	4	3	75%
Measurement and analysis (MA)	8	3	37.5%
Project monitoring and control (PMC)	10	5	50%
Project planning (PP)	14	12	86%
Supplier agreement management (SAM)	6	0	0%
Total	**54**	**35**	**64.81%**

management information needs. This area defines eight specific practices, but only three of them are supported by MDE practices. Table 5 shows the supported specific practices.

Project Monitoring and Control (PMC) Process Area

The purpose of project monitoring and control is to provide an understanding of the project's progress so appropriate corrective actions can be taken when the project's performance deviates significantly from the plan. This area defines 10 specific practices; only five of them are supported by MDE practices.

Project Planning (PP) Process Area

The purpose of the project planning process area is to establish and maintain plans that define project activities. This process area has a high degree of MDE support (86 percent). The only two practices unsupported by MDE are

- SP3.1. Review plans that affect the project
- SP3.2. Reconcile work levels and resource levels

Supplier Agreement Management (SAM) Process Area

This area was excluded from the analysis, since it does not apply to an MDE project. Outsourcing of external products and services is part of the organization's strategy and is beyond its scope.

SUPPORT LIMITATIONS PROVIDED BY MDE TO EACH CMMI-DEV AREA

This section deals with the possible causes and consequences of the limitations in support provided by MDE to each CMMI-DEV area (see Table 6).

Configuration Management (CM) Process Area

This process area has high MDE support (100 percent).

Requirements Management (REQM) Process Area

This process area has high MDE support (100 percent).

Process and Product Quality Assurance (PPQA) Process Area

This process area has high MDE support (75 percent). The practice SP2.2, "Communicate and resolve noncompliance issue," is the only CMMI-DEV specific practice that does not have support. For this specific practice, the CMMI document states that "noncompliance issues

are problems identified in evaluations that reflect lack of adherence to applicable standards, process descriptions or procedures. The status of noncompliance issues provides an indication of quality trends."

Examples of work products indicated in CMMI are corrective action reports, evaluation reports, and quality trends. An important problem in MDE relates to the lack of procedures indicating the need to record the noncompliance issues. For example, MDE practice GP5, "Produce sample artifacts for key scenarios," applies to control of noncompliance issues, but it does not include the need to generate procedures for their record. In this case, the GP5 practice should have a subpractice that states the need to perform corrective action reports and evaluation reports when failures in the sample artifacts are detected.

Measurement and Analysis (MA) Process Area

This constitutes the Level 2 process area with the lowest MDE support: 37.5 percent support. Nonsupported practices are as follows:

- SP1.4. Specify analysis procedures
- SP2.1. Obtain measurement data
- SP2.2. Analyze measurement data
- SP2.3. Store data and results
- SP2.4. Communicate the results

Three out of four specific practices of the specific goal (SG) 1, "Align measurement and analysis activities," are supported. None of the specific practices of the SG2 is supported. In general, when analyzing MDE support to specific practices of goal 1, the authors observe that the support found, though existing, is based on few MDE practices, especially SP1.1 and SP1.2, which are related to the needs of performing measurements (MDE GP23, which refers to the utility of generating metrics after the MDE project).

Following the analysis, the authors observe that specific practice SP1.4 "Specify analysis procedure," of SG1, is not supported by MDE. The practice is oriented to the specification of analysis procedures that allow details on how collected data are analyzed and communicated. According to the CMMI-DEV 1.3, data mean the information recorded that can include technical data, software documentation, financing information, fact representation, numbers, or data of any nature that can be communicated, stored, and processed. In the case of MDE, the possible data would be technical data, software documentation, and other information related to the MDE project.

Therefore, to support the specific practice, the appropriate procedures should be specified for the analysis of the MDE tools. In general terms, although the actions for generating those tools (which are the data in this case) are specified, the MDE practices do not generally indicate the procedures to record and analyze them.

Then, SG2, "Provide measurement results," corresponding to unsupported specific practices SP2.1, SP2.2, SP2.3, and SP 2.4, is explained in CMMI. The primary reason for doing measurement and analysis is to address identified information needs, derived from organization and business goals. In this case, CMMI specific practices refer to the need for obtaining, recording, and storing the results of measurements to obtain information.

As in the case of SP 1.4, recording of results is a weak point in MDE and was also highlighted for the process and product quality assurance process area. This problem should be resolved by originating new MDE practices using expert recommendations.

Project Monitoring and Control (PMC) Process Area

This is the second process area with the lowest MDE support (50 percent support). Unsupported specific practices are as follows:

- SP1.6 Conduct progress reviews
- SP1.7 Conduct milestone reviews
- SP2.1 Analyze issues
- SP2.2 Take corrective actions
- SP2.3 Manage corrective actions

In the authors' opinion, the insufficient MDE support to this process area is due to the fact that the authors, whom this study was based on, state that follow-up of an MDE project is similar to that of any software project, and no practices or recommendations have been determined aiming at the specific matter of follow-up and control of the MDE project. However, which MDE-specific issues should be analyzed, which corrective actions should be conducted throughout the process development, which milestones should be key, and how corrective actions should be managed are some of the questions related to an MDE project that could be analyzed. Previous analysis of the area under study revealed that even for the specific supported practices, few MDE practices have been found.

As an example, the authors can analyze how to improve support to specific practices that have limited support, such as specific practice SP1.5, "Monitor stakeholder's involvement," supported only by MDE practice GP24. In MDE, the most significant stakeholders are business application developers who do not participate in the MDE project but will be its users. Other practices to support it, in addition to practice GP24, should be generated to ensure participation of stakeholders throughout the life cycle of the MDE process.

Project Planning (PP) Process Area

This process area has a high degree of MDE support (86 percent). The only two practices unsupported by MDE are:

- SP3.1. Review plans that affect the project
- SP3.2. Reconcile work levels and resource levels

The MDE GP6 practice indicates that after defining the tool chain, "it is possible to create a detailed plan of the necessary effort to build the MDE tool," but it states "possible" and not "compulsory;" moreover, it is not stated as an independent task, which the authors interpret to be the correct classification. The authors' recommendation would be to include it explicitly as a task in the list of assignments to be conducted in the MDE project. In fact, it is understood that the task of estimating the effort and cost of an MDE project is not given sufficient importance. It must be kept in mind that lack of clarity on the cost of the project is one of the most important obstacles for managers and directors in adopting the MDE.

Although it is generally stated that using MDE can save costs, when an enterprise starts implementing the MDE that is not often true. This is why cost savings should be examined as repositories are built and reuse becomes possible. Moreover, although some existing MDE practices recommend the evaluation of artifact reuse when designed and built (practices GP11, GP22, and GP38), few other specific practices ensure that this will become effective. For example, practice GP2 applies to the reuse in itself when indicating that it is necessary "to ensure that in the following project, tasks that face reuse of MDE artifacts will be included;" however, specific tasks needed to reach this goal are not stated.

SP1.4, "Estimate effort and cost," supported by MDE practices GP6 and GP23, is another issue. Given that CMMI-DEV 1.3 states that when this specific practice is analyzed in the project planning process area, it expresses "estimates of effort and cost are generally based on results of analysis using historical data models;" this is reasonable in the case of MDE, and estimating efforts and costs becomes difficult. This weak point in MDE is that there is insufficient experience regarding use and reuse of MDE artifacts in the organizations.

Overall Summary

The simultaneous analysis of the seven process areas reveals that some CMMI specific practices are supported by only two MDE practices, while others (for instance, SP2.1 from the project planning process area with 34 supporting practices) are strongly supported. Therefore, it seems reasonable to determine a range, in which to say MDE support is weak, absent, or strong. Taking average values of the number of specific practices supported by MDE in a given process area, it is reasonable to take two MDE practices as a limit. This value corresponds to the two process areas with less MDE support: project monitoring and control, and measurement and analysis.

Taking into account the results obtained, the authors establish that if the average number of specific practices supported by MDE is lower than or equal to 2 the support is *weak*; and conversely, if it is higher, the support is *strong*. The summary of this definition is displayed in Table 7.

TABLE 7 Number of CMMI specific practices supported by MDE practices per process area

Process area	Support (#GPs/#SPs)	Type of support
Configuration management (CM)	7.28	Strong
Requirements management (REQM)	3.8	Strong
Process and product quality assurance (PPQA)	5.33	Strong
Measurement and analysis (MA)	2	Weak
Project monitoring and control (PMC)	2	Weak
Project planning (PP)	11.16	Strong
Supplier agreement management (SAM)	0	Not supported

PROPOSAL VALIDATION

As a preliminary validation of the analysis conducted in this study, five software engineers who are specialists in quality management were asked to provide their opinion on the results and conclusions obtained. A survey was constructed to record their agreement or disagreement on each item of the proposal for three Level 2 process areas, and within each area only two of the specific practices were selected. A detailed description of the survey methodology and justification can be found in Esterkin (2014).

Tables 8, 9, and 10 show the support obtained. Each column shows the approval percentage expressed by each professional consulted, while the last column shows the average value obtained for each specific practice. This leads to a final result for the three process areas displayed in Table 11.

The evaluation was only carried out with five professionals since it was very difficult to find experts in CMMI-DEV with knowledge of MDE. But, the authors have noticed that it is more common to find experts in MDE with some knowledge of CMMI-DEV and that this is enough to understand the proposal. The authors are currently improving the evaluation following this approach.

TABLE 8 Results of the survey of the configuration management (CM) process area

SP	Respondents			Avg
	1	2	3	
SP1.1	87.5%	100%	100%	95.8%
SP2.1	100%	100%	100%	100%

TABLE 9 Results of the survey of the requirement management (REQM) process area

SP	Respondents			Avg
	1	2	3	
SP1.1	100%	40%	100%	80%
SP1.5	100%	100%	100%	100%

TABLE 10 Results of the survey of the project planning (PP) process area

SP	Respondents			Avg
	1	2	3	
SP1.1	85.7%	80.9%	38.1%	65.2%
SP1.4	50%	50%	100%	66.6%

TABLE 11 Results of preliminary validation for MDE support

Process Area	MDE Support (validation results)
Configuration management (CM)	Yes, for both SPs surveyed
Requirements management (RM)	Yes, for both SPs surveyed
Project planning (PP)	Yes, for SP1.1; Yes (weak) for SP1.4

RELATED WORK AND RECOMMENDATIONS

In Rios et al. (2006) the authors show how a maturity model developed within one project helped several enterprises to adopt MDE. The "Modelware" project was conducted in Spain between 2002 and 2006 and defined five maturity levels that placed the enterprises adopting MDE in different degrees of MDE practice and artifacts usage. Instead of defining different levels of accomplishment of MDE practices, the authors propose the adoption of a recognized quality model, such as the CMMI, in response to the need for integrating the control of specific development practices into a new paradigm of software development.

Quintero et al. (2012) state the "top" problems of the MDE and introduces recommendations on how to manage and mitigate them. They present 10 problems:

1. Models become out of date and inconsistent with the code.
2. Models cannot be easily exchanged between tools.
3. Modeling tools are hard to install, learn, configure, and use.
4. The code generated from a modeling tool is unsatisfactory.
5. The details that need to be implemented are hard to describe
6. When the modeling tools change, the models become obsolete.
7. Modeling tools are too expensive.
8. Modeling tools do not allow the analysis of my design the way I would like.
9. Modeling tools hide too many details that would be visible in the code.
10. Organizational culture may not like the use of models.

Analysis of Quintero's 10 problems revealed that the first nine are technical matters related to models and the modeling tools used to generate them; the remaining problem is related to the organizational culture and its willingness to use models. However, in some cases, its technical recommendations could help increase MDE support to CMMI Level 2 when incorporating into the analysis the tools used for MDE development, characteristics, and costs. This is the case of practice SP1.4, "Estimate effort and cost" associated with the project planning process area. The recommendation to face problem 7, "Modeling tools are too expensive," is given in the following response: "There are many free software tools; however, if in the selection process, the tool selected is too expensive, the first projects must be profitable in cost, time, and quality to justify the investment." This reveals an additional element that was not considered in MDE: the cost of the MDE tools should be taken into account when estimating the cost of an MDE project. In addition, MDE support is reinforced by SP1.3, SP1.4, and SP1.5 of the requirements management process area.

The authors' study has investigated whether MDE support can be improved by applying those recommendations and shows that the analysis of the specific practices in CMMI Level 2 (considering MDE top problems and the recommendations to face them) introduces few new elements regarding practices unsupported by MDE. However, it reinforces the support with new elements in some specific practices that do support MDE. This is reasonable given that CMMI Level 2 is the "managed" level, and Quintero et al. (2012) analyze the tools from a technical point of view.

In Calic, Dascalu, and Egbert (2008), the authors set out challenges that MDE is still facing, such as tool limitations and the lack of integration to the business process modeling models in the transformation of models. The article recognizes current risks related to both weaknesses; however, it also analyzes and describes a way to mitigate these risks through controls introduced in the notes, which provide suggestions for new elements in the CMMI model. Regarding this subject, the authors' study incorporates a detailed analysis of CMMI-DEV Level 2 practices and their connection with MDE.

In Lins de Vasconcelos et al. (2011) the authors defined a software development process based on MDE from requirements to final code generation that integrates elements of the i* framework and the goal-oriented requirement engineering (GORE) methodology, compatible with CMMI-DEV. They focused on defining a new software development process designed to be compatible with the maturity model. The authors' proposal is not focused on any process in particular, but on MDE methodology in general; thus, it attempts to be applicable to any MDE development process.

Finally, in SEI (2016), McMahon (2011), and Konrad and McGraw (2008) the connection of the CMMI-DEV model with agile methodologies is described, specifying the key aspects that allow the coexistence of both approaches. Although these studies focus on another software development paradigm, it is convenient to take them into account, since they provide the basis for the mapping definition between the CMMI-DEV and a software development paradigm.

CONCLUSIONS

According to this study, MDE provides a high degree of support for the configuration management, process and product quality assurance, and project planning process areas of CMMI-DEV 1.3. In general terms, the authors can conclude that in MDE detailed description of procedures, documentation, follow-up methods, and other topics are still missing. Problems related to lack of support are still unresolved in process areas with low MDE support, such as measurement and analysis, and project monitoring and control; these should be addressed in continued MDE development to improve support of CMMI-DEV Level 2.

Although the unsupported (or weakly supported) process areas and specific practices are explained to a large extent by the recent introduction of MDE into software development in industry, the present study aimed to identify the gaps that should be resolved to stimulate the use of MDE in organizations.

Software developers who apply the MDE need a standard that defines the guidelines and good practices, taking into account the risks and particularities of the MDE. On the other hand, CMMI must offer support to this growing sector of development teams and companies that decide to incorporate new paradigms of software development. However, for this integration to occur, it is necessary for researchers, software developers, MDE toolmakers, and the CMMI institute to coordinate efforts to interpret CMMI-DEV practices within an MDE development process.

This would include at least the following activities:

- Take into account that there are risks that could affect the quality of the product, in a

software development project, that are not mitigated if only the practices of the MDE paradigm are applied.

- Complement MDE with other methodologies, such as the rational unified process, as described in Eeles (2004) and Balmelli et al. (2006).
- Document and disseminate the MDE good practices as part of the CMMI-DEV model.
- Define the practices to be evaluated in the MDE process.
- Consider the inclusion of additional artifacts in the MDE modeling to improve compliance with CMMI-DEV, especially for unsupported (or weakly supported) process areas.
- Document good MDE practices and additional artifacts in the process asset library (PAL) of the organization. The PAL is a repository of information used to keep and make available processes that are useful for those who are defining, implementing, and managing processes in the organization (Garcia 2004).
- Provide special training on MDE to the CMMI evaluators who evaluate a company that develops under MDE.

One of the primary limitations of this study was the use of a small expert panel for preliminary validation. A further study, which collects proposals and MDE practice recommendations from a much larger group of specialist professionals in software engineering, is planned to help improve the findings and contribute to increased support for CMMI-DEV Level 2.

REFERENCES

Balmelli, L., D. Brown, M. Cantor, and M. Mott. 2006. Model-driven systems Development. *IBM Systems Journal - Model-Driven Software Development* 45, no. 3.

Brambilla, M., J. Cabot, and M. Wimmer. 2012. Model-driven software engineering in practice. San Rafael, CA: Morgan & Claypool Publishers.

Calic, T., S. Dascalu, and D. Egbert. 2008. Tools for MDA software development: Evaluation criteria and set of desirable features. Fifth International Conference on Information Technology: New Generations, 44-50.

Di Ruscio, D., R. F. Paige, and A. Pierantonio, eds. 2014. Science of Computer Programming. Special issue on Success Stories in Model Driven Engineering 89:69-222. Elsevier.

Eeles, P. 2004. MDA and RUP. IBM Software Group. IBM Corporation. Available at: http://www.architecting.co.uk/presentations/MDA and RUP.pdf.

Esterkin, V. 2014. Análisis y Evaluación del MDE (model driven development) desde la perspectiva de CMMI DEV 1.3 Nivel 2. Master thesis. Universidad Abierta Interamericana. Buenos Aires, Argentina.

Garcia, S. 2004. *What is a process asset library? Why should you care.* Boston, MA: Aimware Professional Services Inc.

Kleppe, J. Warner, and W. Bast. 2003. *MDA explained. The model driven architecture: Practice and promise.* Boston, MA: Addison-Wesley.

Konrad, M., and S. McGraw. 2008. *CMMI & Agile*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Lins de Vasconcelos, A., G. Giachetti, B. Marín, and O. Pastor. 2011. Towards a CMMI-compliant goal-oriented software process through model-driven development. *The Practice of Enterprise Modeling*, 253-267. Lecture Notes in Business Information Processing. New York, NY: Springer.

McMahon, P. E. 2011. *Integrating CMMI and agile development.* Boston, MA: Addison Wesley.

Object Management Group. 2015. OMG success stories. Available at: www.omg.org.

Pons, C., R. Giardini, and G. Pérez. 2010. Desarrollo de software dirigido por modelos. Editorial Edulp and McGraw Hill Educación, 279. Available at: http://sedici.unlp.edu.ar/handle/10915/26667.

Quintero, J., P. Rucinque, R. Anaya, and G. Piedrahita. 2012. How to face the top MDE adoption problems: An exploratory study case (CLEI). XXXVIII Conferencia Latinoamericana, 1-10. October, Medellín, Colombia.

Ríos, E., T. Bozheva, A. Bediaga, N. Guilloreau. A. Rensink, and J. Warmer. 2006. MDE maturity model: A roadmap for introducing model-driven development. In ECMDA-FA 2006. LNCS 4066, 78-89.

SEI. 2010. CMMI-DEV 1.3. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. Available at: http://www.sei.cmu.edu/reports/10tr033.pdf.

SEI. 2016. *A guide to scrum and CMMI: Improving agile performance with CMMI.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Stahl, T., and M. Voelter. 2006. Model-driven software development. *Technology, engineering, management.* New York, NY: Wiley.

Swithinbank, P., M. Chessell, T. Gardner, C. Griffin, J. Man, H. Wylie, and L. Yusuf. 2005. Patterns: Model-driven development using IBM Rational software architect. Available at: http://www.redbooks.ibm.com/redbooks.

BIOGRAPHIES

Claudia Pons is a full professor at the University of La Plata and the director of the Center of High Studies in Computer Technology (CAETI) in Buenos Aires, Argentina. She obtained a doctorate in the application of formal methods to object oriented modeling in 2000. She has participated in several research and development projects and has been part of the program committee of the ACM/IEEE MoDELS Conference and other international conferences. She has published papers and books in this field of knowledge. Pons can be reached by email at claudia.pons.33@gmail.com.

Magister Viviana Esterkin obtained a master's degree at the Universidad Abierta Interamericana (UAI) in 2014. She works as project evaluator and CMMI assessor, and currently she is a professor at the University of Tres de Febrero (Untref), Buenos Aires, Argentina.