Software Process Management: A Model-Based Approach

L. García-Borgoñon, J.A. García-García, M. Alba,and M.J. Escalona

Abstract Business processes constitute one major asset in an organization and software businesses are not an exception. Processes definition, maintenance, and management are key aspects to control and define how to build software systems up and also to support decision-making. In this paper, a model-based approach is proposed to facilitate these processes. Thus, a global environment for business processes in software development is presented. The final results are illustrated through the NDTQ-Framework, a solution based on this approach that is currently being used in software development organizations.

1 Introduction

Since processes are recognized as fundamental asset in organizations, there is always an evolving interest to define, document, manage, and improve them. The promise of achieving better quality and greater efficiency and effectiveness in the cost and effort resulting from product development has involved the adoption of processes in several domains, some of which have reached a certain maturity level in this field. However, this is not the case of Software and System Processes, which is still in its early usage days.

In the last years software organizations are using Business Process Management (BPM) as a mechanism to control and define how to build software systems up.

Different techniques for business process modeling and business process execution, as well as for their relationship, have been proposed. Business Process Modeling Notation [23] has become a widely used standard in process modeling environment, and Business Process Execution Language (BPEL) [24] has consolidated as the language for business process execution.

Software Process Engineering has been established as an independent research area from general business process. Its main objective deals with improving software development practices by proposing (a) better ways for designing the organization processes and (b) better ways for improving these processes both individually and as a whole.

In order to efficiently define and execute software development processes, it should be necessary to establish (1) a process modeling language rich enough so as to define all main aspects of software processes, (2) an easy-to-use process modeling environment that is flexible enough for different project categories, and (3) a process execution environment easy to be integrated into existing development tool chains [25].

The evolution of software process modeling has been studied during the past decades. A myriad of process languages and models have been developed; however, software process models are not executable and there are few process execution environments.

In the last years, the model-driven engineering (MDE) [26] has been established as a usual approach for software development [27], what has shaped the software industry to be model centric. Its aim is development through the evolution of one model from requirement until deployment by means of a series of transformations, which can progressively be achieved through coherence between software process modeling and software development paradigm [28].

This paper evaluates how a model-based approach can make easier the BPM in information system development organizations, and it also illustrates a practical example that uses this approach for software process definition, maintenance, and improvement.

The paper is structured as follows: Section 2 shows the main work related to software processes. Section 3 introduces the proposed metamodel, and Sect. 4 presents the Unified Modeling Language (UML) profile, which is used to integrate the metamodel into a tool. Section 5 analyzes a global environment for software process management. These results are illustrated in the NDTQ-Framework, a solution based on this approach that is currently being used in software development organizations. Finally, Sect. 6 outlines conclusions from these studies as well as proposes future work in these lines of research.

2 Related Works

The first section of this related work refers to different proposals that are referenced in the organizational processes definition as guidelines.

A process is defined as the set of partially ordered steps or activities, with sets of related artifacts, human and computerized resources, and organizational structures and constraints intended to produce and maintain the requested software deliverables [1]. Support processes have been developed in order to facilitate software organization activities, different standards, methodologies, and methods focused on management, development, evaluation, and software life cycle and organizational life cycle. At this point, it is necessary to highlight International Organization for Standardization (ISO) standards that prescribe processes, each of them with a specific aim:

- ISO/IEC 122007:2008 [2] establishes a common framework for software life cycle of processes with well-defined terminology that can be referenced by the software industry.
- ISO/IEC 15288:2008 [3] establishes a common framework for describing human-created life cycle of systems. It determines a set of processes and associated terminology that can be applied at any level in the hierarchy of a system's structure.
- ISO/IEC TR 24744:2007 [4] was defined through the large number of standards with similar concepts used for describing process reference models whose process descriptions vary in format, content, and level of prescription. Uniform descriptions combine processes from different reference models, facilitating the development and comparison of new models.

The second section of this related work presents the most popular languages and notations to process definitions.

The usual comparison between software processes and manufactured processes has entailed many efforts to describe and automate them. Thus, these efforts have been addressed in different stages. First-generation languages, known as Software Process Modeling Languages (SPMLs), were developed during the 1990s. Some of them were rule based such as MARVEL [5]; others were Petri net-based such as SPADE [6], or some others programming language-based such as SPELL or APPL/A [7]. All of them were focused on executability, but their complexity and emphasis on formality and inflexibility have made them not to succeed in the industry.

An alternative is BPMN, since it still remains as the preferred technology in the industry. Its simplicity, standardization, and support for executing processes are the key for being widely used. However, this language is more oriented towards business processes description, which constitutes a less specific scope than this of software processes.

As a result, many UML-based approaches were developed and a new language generation for software processes was introduced. Some of them were UML 1.3 based such as in Di Nitto et al. [8]; another uses a subset of UML 1.4 such as Chou's approach [9]. In addition, UML4SPM [10] was proposed as a candidate for the new version of the Object Management Group's (OMG) standard called Software Process Engineering Metamodel (SPEM) [11]; nevertheless it is based on SPEM 1.1 and UML 2.0 behavior modeling concepts. It mainly focuses on the enactment support and two alternatives were defined.

From the standardization point of view and regarding the software-specific domain, there are two main languages today: ISO/IEC 24744 [12] and SPEM 2.0. As it can be noticed, both pursue the same objective despite they differ in some aspects:

- ISO/IEC 24744, Software Engineering Metamodel for Development Methodologies, is an international standard that defines a metamodel for methodologies of development in the software environment. It does not use OMG's strict metamodeling approach, but the power-type pattern that was adopted for metamodeling in the methodologies domain in [13].
- SPEM 2.0 provides a language for software methodologies, takes the Meta-Object Facility (MOF) [14] as a starting point, and is defined as a UML profile. It has a very difficult structure since it introduces extension mechanisms, compliance points, and concepts to distinguish method contents from processes, what make the specification turns out very complex and difficult to understand.

In recent years, model-based engineering has been established as a standard approach for software development. MODAL (Model-Oriented Development Application Language) [15] is a SPEM 2.0-based process modeling language that introduces additional concepts to exploit the potential of MDE. Unlike SPEM, it is more focused on process model execution, even though the standard complexity is reproduced here.

To summarize, efforts in Software Process Engineering area have been headed by two different aspects: on the one hand, methods and standards definition in order to prescribe what a process should accomplish and, on the other hand, the need to have a language for the process definition. Last decades have witnessed the birth of many approaches and a parallelism between software development paradigms and processes for their development has always been evidenced. As a result, proposing MDE for process engineering may be a solution to cope with this classic problem.

3 A Metamodel for Software Process Management

Many approaches have been developed in order to recommend the required elements in a process as well as describing it. The main element in all of them is Software Process, but the concepts included in these approaches differ. This situation has motivated the development of the standard ISO/IEC TR 24744, which is issued as a guideline for the process description.

These and the fact of using MDE to possibly manage the conceptual complexity of Software Process Engineering have been the basis of our proposal, that is, a metamodel for Software Process Support. This approach is presented in the form of a MOF-compliant metamodel as it is shown in Fig. 1.

The *Process* metaclass is the main class in the metamodel. It represents a set of ordered actions executed by someone in order to produce something. The attributes in this metaclass, the name of the process and a short description of it, have been incorporated in accordance with ISO/IEC TR 24744 standard.

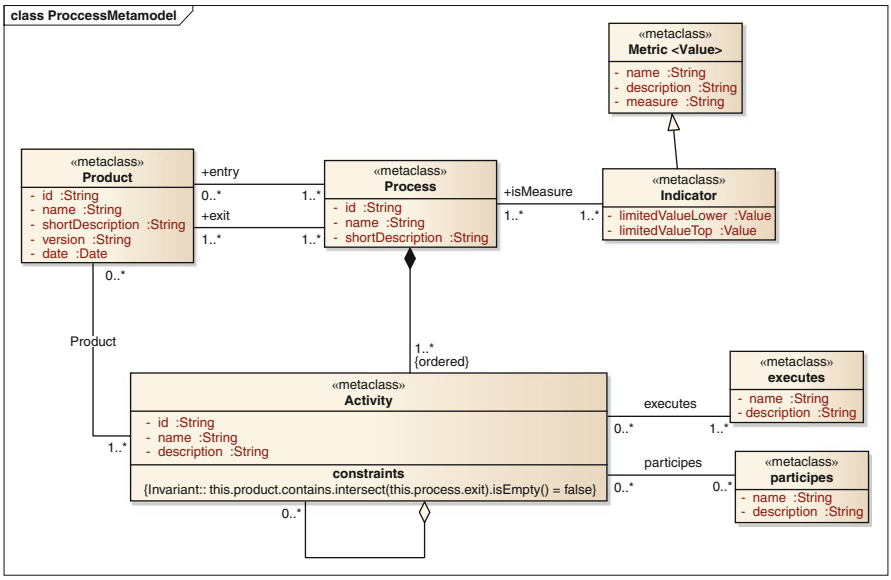

Fig. 1 Process metamodel

The actions included in a process are represented by the *Activity* metaclass. These activities are arranged within the process, although one activity can also contain several activities. This recursive relation is explicitly requested by the standard.

There are two kinds of stakeholders involved in performing an activity, which are classified depending on the involvement degree: an actor, who is the main executor of the activity, is represented by the *Executes* metaclass, whereas the *Participates* metaclass represents the set of stakeholders who contribute, but is not directly the main responsible for it. Each of them includes a name and a description attributes. At least one executor is necessary to define an activity.

The *Product* metaclass represents the product resulting when executing a process. This product can be developed from scratch, during the execution of the activities, or can be provided from a previous product. In this case, the original product will be defined as an entry and modified within the activity in order to obtain the outcome.

Finally, the *Metric* metaclass deals with process information elements. Metrics in processes are limited-value established and represented through the *Indicator* metaclass.

The main feature of our approach is simplicity, unlike others presented in the previous section. This metamodel offers a suitable mechanism for process definition by covering the main software process concepts and being an ISO/IEC TR 24744 compliant. This simplicity must develop a whole model-based solution, as it will be studied in next sections.

4 An Enterprise Architecture Profile

A profile is defined in a UML package through the stereotype <<profile>>, indicating that it will extend a metamodel. There are three mechanisms used to define these profiles:

- Stereotypes. They are defined by a name and a series of elements of the metamodel with which they can be associated. Graphically, the stereotypes are set in boxes, <<stereotype>>.
- Restrictions. They impose conditions on the previously stereotyped elements of the metamodel, so as to describe, among others, the conditions that they have to check in a "well-built" model. A commonly used language of restriction is OCL.[1]

  e.j: Invariant:: this.product.contains.intersect(this.process.exit).isEmpty()=false

- Tagged Values. There are additional meta-attributes that can be associated with the metamodel of a metaclass extended through a profile.

To build the profile, the software process metamodel should be used as shown in Fig. 1. A stereotype is included within the package <<profile>> for each element of the metamodel contained in the profile. It is named as the metamodel elements; thus a relationship between the metamodel and the profile is established. Then, any item needed to define the metamodel can be labeled with a stereotype [29]. In this case "Participies," "Executes," "Activity," "Product," "Process," "Indicator," and "Metric" will be created.

In our example, Participies and Executes extend of the metaclass "Actor;" Activity and Process extend of the metaclass "Activity;" and Product, Metric, and Indicator will extend of Artifact. Tagged values are defined as the profile attributes elements that appear in the metamodel. The definition of their types and possible initial values must be included.

In the given example, the attribute "measure" of the metaclass Metric and the attributes LimitedValueTop and LimitedValueLower of the metaclass Indicator have to be added as tagged values since the metaclass implicitly has other attributes. We will define the profile constraints from the domain constraints. For example, either the multiple associations listed in the domain metamodel or any individual's business rules of the application must be translated when the relevant restrictions are defined. In this case, there is one the invariant of Activity (Fig. 2).

Invariant:: this.product.contains.intersect(this.process.exit).isEmpty()=false

5 An Example: NDTQ-Framework

NDTQ-Framework is based on the metamodel analyzed in Sect. 3. It formally defines all processes currently supported by NDT [16], although it is also flexible and can be adapted to different levels and typologies of developments.

[1] http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL.

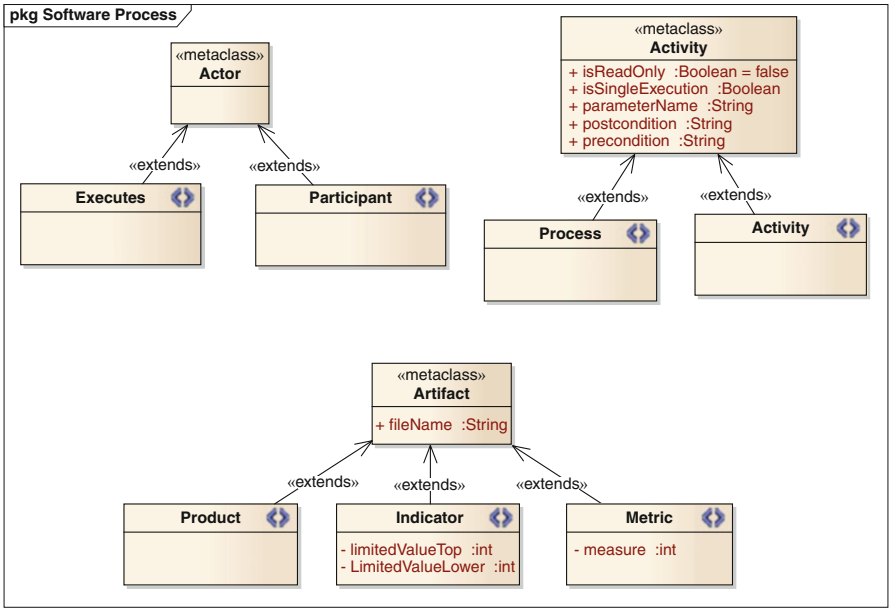

Fig. 2 Software process profile

NDTQ-Framework defines processes on the Enterprise Architect tool through a specific pattern dealing with process description, based on the proposed ISO/IEC TR 24774:2007 and ISO12207. It thoroughly defines the six groups of processes that include NDT-methodology. However, due to their extension, they will not be fully explained in this paper, but briefly introduced below. Only the requirements engineering process, which belongs to the software development processes, will be pointed out.

- Processes of Software Development. These processes support different types of life cycles: classical or sequential, agile, iterative, and incremental. They are defined on the basis of NDT life cycle, although the terminology has been referenced on ISO 12207 standard.
- Processes of Software Maintenance. These processes are based on the best practices defined both in ITIL [17] and CMMI [18]. This group only defines the process beginning when the project is in production and ending when the system falls into disuse.
- Processes of Testing. These processes are based on the first results of ISO/IEC 29119 [19]. This group defines Testing Organization, Testing Management, and Testing Execution.
- Processes of Software Quality. This group of processes is based on ISO 9001:2008 and the good practices of CMMI. This group defines the following processes: Managing Corrective and Preventive Actions, Documentation, Control and Records, Human Resource Management, Customer's Satisfaction, Data Analysis and Review by the Director of the Organization, Technology Watch, Monitoring Indicators, Elaboration of Standards, and Internal Audits.

- Processes of Project Management. They are based on some of the practices of the Project Management Body of Knowledge (PMBOK) [20] and CMMI. This group defines the following processes: Event Management, Personnel Management, Project Monitoring, Change Management, Schedule Management, and Cost Management.
- Processes of Security. This process is based on ISO 27001. This group defines the following processes: Physical and Environmental Security, Asset Management, Risk Analysis, Security Organization, Communications Management and Operations, Security Incident Management, and Business Continuity Management.

The following information is provided by each of the processes mentioned above: roles or participants involved in its execution, indicators, tasks or activities, and deliverables of the process. All this information matches with the attributes included in the metamodel defined in Sect. 3. To illustrate all the processes above is out of the scope of this paper. However, in order to present the approach, the requirements engineering process will be explained in detail. Figure 3 shows the map of activities for the requirements engineering process.

The Requirements Engineering includes the necessary flow of activities to generate the system requirements document. NDT-Suite, which consists of a set of very useful free tools to apply the NDT-methodology, can automatically create and revise the requirement documents [21].

After finishing this process, all participants must reach a consensus and approve the final system requirements document. The requirements engineering process involves four roles or participants: Project Manager, Monitoring Committee, Project Manager at SQA, and Responsible for User's Area.

The Project Manager must carry out the first activity within the requirements engineering process. This activity is referenced in Fig. 3 as "RS01-Get information about the environment and define objectives." In this activity, the Project Manager must approach to the environment where the system will be implemented. During this activity, the terminology used in the project, users and customers who will participate, as well as the main objectives must be set.

In the next activity, the Monitoring Committee must approve the project scope and objectives previously established. This activity is referenced in Fig. 3 as "RS02-To approve the scope." Once the objectives have been identified, the system requirements must be captured. This task is carried out by the Project Manager and includes five activities: (1) identifying and defining storage requirements, (2) identifying and defining actors, (3) identifying and defining functional requirements, (4) identifying and defining interaction requirements, and (5) identifying and defining nonfunctional requirements.

After completing the system requirements document and before being reviewed by the user, the document must be validated by the Project Manager at SQA. This validation involves three other steps: automatic validation through NDT-Suite, which will generate the corresponding report; validation of the technician responsible of SQA; and validation of the technical coordinator.

The Responsible for User's Area, after identifying and describing requirements, must validate it. Audits, thesauri, or ontologies are techniques for requirements

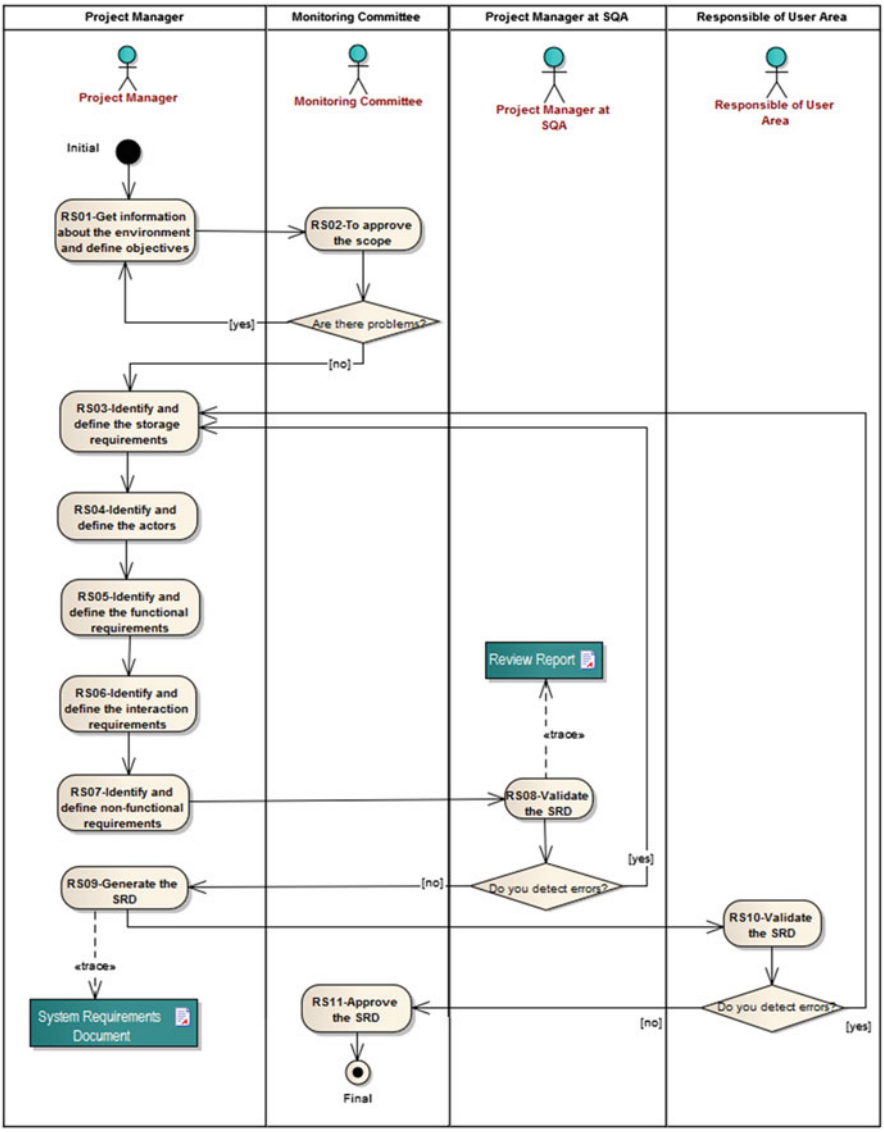

Fig. 3 Map of activities of the requirements engineering process

validation. The final aim of this task is to detect and correct as much errors found during the description of requirements as possible. To elaborate the glossary, NDT-Glossary [22] is recommended (this tool is included in NDT-Suite). Finally, if the Responsible for User's Area has not detected any error or inconsistency in the requirements identification, the Monitoring Committee approves the system requirements document.

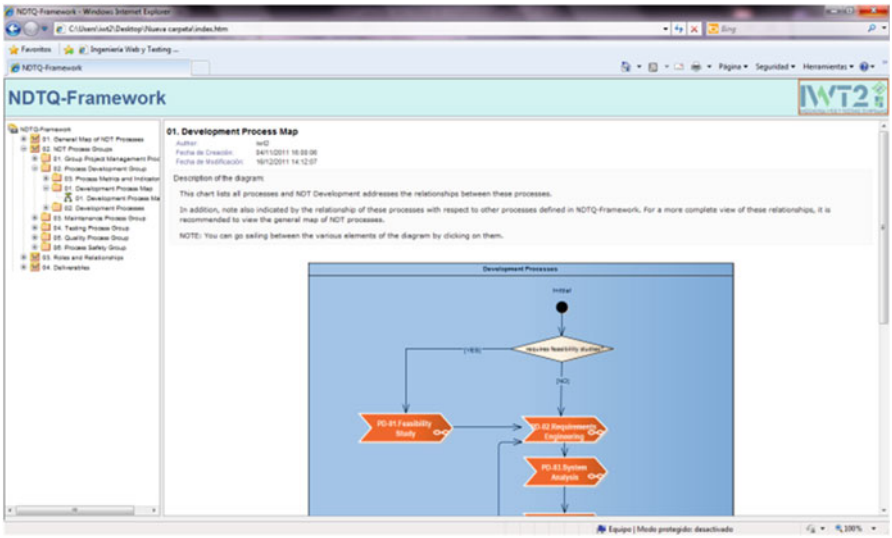

Fig. 4 NDTQ-Framework screenshot

As it has been discussed in this section, each process defined by the NDT-methodology needs to indicate certain relevant information such as the participants involved in its execution, indicators, tasks or activities, and deliverables of the process.

NDTQ-Framework allows quick and direct access to all information associated with an element within the NDT-framework. For instance, from the process, it can be accessed, among others, to all the information related to actors involved in its execution, its associated indicators, its deliverables, or its associated tasks or activities. Besides, NDTQ-Framework guarantees the traceability among these elements. It also offers a set of tools to orchestrate all NDT processes. Today, we are currently working on this line of research (Fig. 4).

6 Conclusions and Future Works

This paper presents a solution for software process definition founded on a model-based approach according to ISO/IEC TR 24744, the standard guideline to establish the concepts related to software processes. This solution is offered by a metamodel and an UML profile and is implemented in Enterprise Architect. A concrete solution named NDTQ-Framework is also presented.

It has been used in several real projects where some relevant conclusions can be deduced. Firstly, a model-based mechanism to define software processes can be very useful, but, if concrete syntaxes and semantic to represent them are not provided, it fails to be used in companies. Otherwise, communication problems can arise.

UML profiles and a UML-based tool seem to be good options to represent them in the software process environment since the development team usually knows this notation.

A tool supposes a required and essential necessity to offer a solution for model process definition. In fact, defining a process under a metamodel guarantees uniformity and a correct definition according to the standard. However, if a suitable tool is not defined, the maintenance of these processes can result too complex, and inconsistencies between the defined process and the real process can arise.

One of the most important aspects concerning this process, which is widely recommended in many standards and good practices manuals, is the continuous improvement. To have a process map, a clear relationship between activities and mechanisms for metric and definition measurement are elements to be taken into account in a continuous improvement program. Consequently, to have a suitable mechanism for defining and maintaining becomes necessary, and it can only be obtained through a tool.

The solution proposed in this paper entails that, in NDTQ-Framework, processes are not only defined under the standard, but they are also connected and interrelated; thus the mechanism to maintain them improves. Besides, this interrelation and connection and the fact that it is based on UML notation reduce the learning curve.

As future work, this approach is aimed for improving in different ways. Firstly, we are working on extending our framework with new processes, like some fragments of ITIL or PMBOK. Secondly, this tool can be improved with a mechanism of orchestration oriented towards the idea of NDTQ-Framework as a whole solution in order to process definition, documentation, and maintenance.

Nevertheless, at this point, there is not support to processes execution, and companies usually have a manual mechanism to solve this situation. In this sense, NDT-Suite can offer a first step to support it, although it is not enough. The processes execution defined by our metamodel represents a very relevant line of research.

Additionally, getting metrics and indicators during the process execution poses another line of research. A solution may allow organizations to identify, extract, and analyze data to support decision-making.

Acknowledgements This research study has been supported by the Tempros project (TIN2010-20057-C03-02) and Red CaSA (TIN 2010-12312-E) of the Ministerio de Ciencia e Innovación, Spain, and NDTQ-Framework project of the Junta de Andalucía, Spain (TIC-5789).

References

1. Lonchamp J (1993) A structured conceptual and terminological framework for software process engineering. In: Proceedings of the 2nd international conference on the software process continuous software process improvement, pp 41–53
2. ISO/IEC, ISO/IEC 12207:2008 (2008) Systems and software engineering – software life cycle processes. International Organization for Standardization
3. ISO/IEC, ISO/IEC 15288:2008 (2008) Systems and software engineering – system life cycle processes. International Organization for Standardization

4. ISO/IEC, ISO/IEC TR 24744:2007 (2007) Software and systems engineering – life cycle management – guidelines for process description. International Organization for Standardization
5. Kaiser G, Barghuti N, Sokolsky M (1990) Preliminary experience with process modeling in the marvel SDE kernel. In: Proceedings IEEE 23th Hawaii ICSS software track
6. Bandinelli SC, Fuggetta A, Ghezzi C (1993) Software process model evolution in the SPADE environment. IEEE T Software Eng 19(12):1128–1144
7. Conradi R, Jaceheri M, Mazzi C, Nguyen M, Aarsten A (1992) Design, use and implementation of SPELL, a language for software process modeling and evolution. Software Process Technology, pp 167–177
8. Di Nitto E, Lavazza L, Schiavoni M, Tracanella E, Trombetta M (2002) Deriving executable process descriptions from UML. In: Proceedings of the 24th international conference on software engineering (ICSE 2002), Compendex, pp 155–165
9. Chou S-C (2002) A process modeling language consisting of high level UML-based diagrams and low Level process language. J Object Technol 1(4):137–163
10. Bendraou R, Gervais M-P, Blanc X (2006) UML4SPM: an executable software process modeling language providing high-level abstractions. Enterprise distributed object computing conference 2006 EDOC 06 10th IEEE International, vol. 6, no. 511731, pp 297–306
11. OMG (2008) SPEM, software & systems process engineering metamodel specification. http://www.omg.org/spec/SPEM
12. ISO/IEC, ISO/IEC 24744:2007 (2007) Software engineering – metamodel for development methodologies. International Organization for Standardization
13. Henderson-Sellers B, Gonzalez-Perez C (2005) The rationale of powertype-based metamodelling to underpin software development methodologies. In: Proceedings of the 2nd Asia-Pacific conference on conceptual modelling, vol. 43, pp 7–16
14. OMG (2011) MOF, meta object facility. http://www.omg.org/spec/MOF/2.4.1
15. Koudri A, Champeau J (2010) MODAL: a SPEM extension to improve co-design process models. In: Proceedings of the 2010 international conference on new modeling concepts for today's software processes: software process, vol. 6195, pp 248–259
16. Escalona MJ, Aragon G (2008) NDT. a model-driven approach for web requirements. IEEE T Software Eng 34(3):377–390
17. ITIL, Information technology infrastructure library. http://www.itil-officialsite.com
18. Chrissis MB, Konrad M, Shrum S (2003) CMMI: guidelines for process integration and product improvement. Addison Wesley, Reading, MA, p 688
19. ISO/IEC, ISO/IEC 29119 Software engineering – software testing standard. International Organization for Standardization
20. Project Management Institute (2008) A guide to the project management body of knowledge (PMBOK® guide) – fourth edition. Project Management Institute, Newtown Square, PA, p 459
21. (2011) NDT-Suite. www.iwt2.org
22. García-García JA, Cutilla CR, Escalona MJ, Alba M (2011) NDT-glossary. A MDE approach for glossary generation. In: Proceedings of the 13th international conference on enterprise information systems. ICCEIS
23. OMG (2011) BPMN, business process modeling notation, Version 2.0. http://www.omg.org/spec/BPMN/2.0/
24. OASIS (2007) WS-BPEL, Web services business process execution language, Version 2.0. http://www.oasis-open.org/standards#wsbpelv2.0
25. Ellner R, Al-Hilank S, Drexler J, Jung M, Kips D, Philippsen M (2010) eSPEM – a SPEM extension for enactable behavior modeling. In: Kühne T, Selic B, Gervais M-P, Terrier F (eds) EdsModelling foundations and applications, vol 6138. Springer, Berlin, pp 116–131
26. Ardagna D, Ghezzi C, Mirandola R (2008) Rethinking the use of models in software architecture. In: Becker S, Plasil F, Reussner R (eds) Quality of software architectures models and architectures, vol 5281. Springer, Heidelberg, pp 1–27
27. Schmidt DC (2006) Model-driven engineering. Computer 39(2):25–31
28. Van Der Straeten R, Mens T, Van Baelen S (2009) Challenges in model-driven software engineering. In: Models in software engineering. Lect Notes Comp Sci 5421:35–47
29. Fuentes L, Vallecillo A (2004) Una introducción a los perfiles UML. Novática 168:6–11