PMLite: An Open Source Solution for Process Monitoring

Alberto Colombo, Ernesto Damiani, and Fulvio Frati
Department of Information Technology - University of Milan
via Bramante 65, 26013 Crema (CR) – Italy
{colombo, damiani, frati}@dti.unimi.it

Abstract. Process Monitoring represents a big challenge for organizations that aim to manage software projects adopting different development paradigms. In fact, across-process enterprise-level measurement campaigns can be difficult to enact since process attributes to retrieve are semantically diverse and may be difficult to integrate. In this paper, we present PMLite (Process Monitoring Lite) an open source solution to this problem. PMLite is based on an open metamodel and paves the way to the definition of ad-hoc open monitoring frameworks.

Keywords: open source, process monitoring, PMLite, open metamodel

1. Introduction

Adopting multiple development processes is becoming common in an increasing number of organizations and communities. Different commercial agreements, or different development scenarios, lead to the adoption of a paradigm rather than a different one; as for instance, a development community could use an agile process to develop an open source Enterprise Resource Planning application, where a commercial software house working for a government agency would most probably follow structured Uniform Process (UP) or a waterfall-like process, often formalized in the supply agreement itself.

Such a situation suggests a new vision about software process monitoring: managers need to have a global view of performance, although development activity may be based on different processes that, at a first sight, are incomparable and whose performance data are hard to integrate.

Many research works have attempted a formalization of the notion of software development process and of the associated measurement framework. Piattini *et al.* [7] describes the advantages of using MOF and XMI to represent development processes, giving an overview of MOF and XMI languages and an example of repository for software development process, while Ventura Martins *et al.* [5] presents the *ProjectIT Initiative*, that provides a complete software development workbench and shows an example of development process metamodel. All the approaches above are related to the SPEM (Software Process Engineering Metamodel) specification [4]

Please use the following format when citing this chapter:

Colombo, A., Damiani, E. and Frati, F., 2008, in IFIP International Federation for Information Processing, Volume 275; *Open Source Development, Communities and Quality*; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 57–68.

proposed by Object Management Group (OMG), that describes a concrete software development process or a family of related software development processes.

Other existing standard frameworks, such as UML and CWM (Common Warehouse Model) have been used to generate metadata describing complex systems, and can be used for development process representation as well.

Starting from these standards and research works, our group has formalized and tested a metamodel [2] for measuring and assessing generic development process models (see Section 2). In this paper, we highlight the progress of our researches presenting the open source application PMLite (Process Monitoring Lite) [6], available on SourceForge, that fully embraces our methodology. The tool has been developed as a proof-of-concept of our approach, and could be adopted in small medium enterprises that need a lightweight across-process monitoring tool.

The paper is organized as follows. Section 2 provides an overview of the metamodel that defines the structure of the tool, whereas Section 3 describes in details PMLite implementation. Finally, Section 4 shows future extensions of our work and Section 5 draws our conclusions.

2. Defining a Metamodel for Process Monitoring

The first step to define a common environment, for measuring different development process, is to produce a general schema, a *metamodel*, that will describe the underlying structure of processes and the relative measuring framework. We start from the work of Piattini *et al.* in [7], that uses MOF and XMI to represent software processes. In particular, MOF (Meta-Object Facility) [3] is a standard supported by OMG [5] that defines a generic pattern for the construction of systems based on metadata. MOF can be described by its four levels structure: starting from the top there is *i)* the definition of all the concepts and attributes of the language itself, used to build a *ii)* metamodel, that defines the structure and semantic of the metadata related to a generic environment; then, this metamodel is used to create *iii)* models, that depict specific objects and describe the structure for *iv)* the user data.

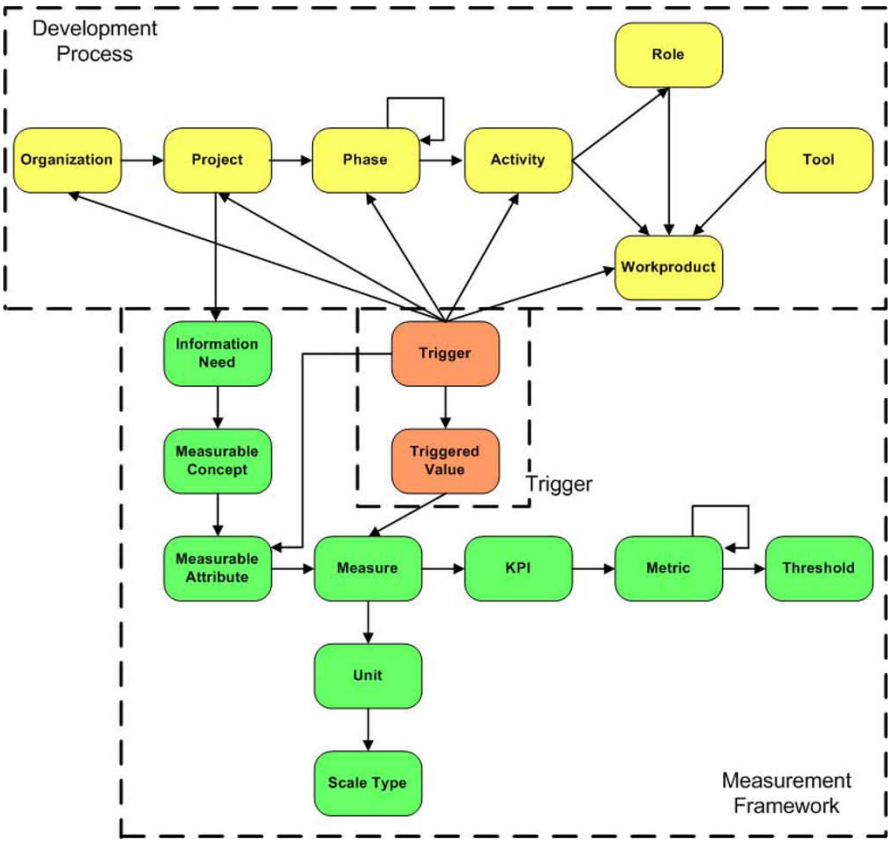

Fig. 2.1 Our modular meta-model. The yellow, green, and red colors correspond to Process, Measurement, and Trigger modules.

Following the MOF approach, we define the development process and the measurement framework metamodels that are used as basis to model specific development processes and measurement frameworks, and a simple trigger layer to connect the two metamodels. The whole metamodel is presented in Fig. 2.1 where the colors define the individual models.

A complete description of any elements of the metamodel could be found in [2]; for the sake of conciseness, in this paper we limit the description only to the elements that would be directly involved in PMLite development.

Metamodel description.

The three colors in Fig. 2.1 define the three parts of our modular metamodel. It is important to note how the process module is independent from the measurement one, thanks to decoupling via the trigger layer. Such decoupling allows to apply the same measurement framework to projects implementing different development processes, and, consequently, to elaborate across-process assessment.

The yellow blocks identify the development process module. The first node is the element *Organization*, that describe the overall organization and allows enterprise-level measurement campaigns. Each organization manages a set of *Project* classes, each one realized by its own set of *Phases*, that are characteristic of a particular development paradigm. Each phase has its own set of *Activity* nodes, that effectively describes the task put in action during whole process. Furthermore, each phase could be linked to another phase to describe iterative models.

The green module describes the structure of the measurement framework. The framework is based on the *Goal-Question-Metric* (GQM) approach [1], which drives the creation of a measurement system starting first, from the identification of the *goals* of the measures, then of the *questions* that will evaluate them through a set of specific *metrics*. The first element, *Information Need*, is the container node of the module and describes the information need over which the measurement is based and it is used as conceptual link for the two modules. Then, following the GQM approach, the entity *Measurable Concept* defines the areas, i.e. goals, over which the analysis is based. The *Measurable Attributes* node defines the attributes to measure in order to accomplish analysis goals. Further, this element provides the way how these attributes could be gathered; indeed, there is a strict relation between *Work Product* and *Measurable Attribute* classes, since the latter are attributes that could be extracted from the former. The *Measure* class describes the value of the measured attributes and it is strictly related to *KPI* (Key Performance Indicator) and *Metric* nodes, that define an elaboration of the measure instances in order to provide indicators that, respectively, lower the cardinality of the measures and qualitatively evaluate the results.

Finally, the red module isolates the trigger representation, which simply defines the *Trigger* entities, i.e. a plug-in, that physically extracts the attributes values from the work products, storing data in the *Triggered Value* class. As said above, triggers allow modules to be independent one from each other, since they know which attributes to extract and in which work product they have to be physically extracted and in which way.

Instance example: the Scrum agile process.

Our open metamodel has been designed as general as possible, in order to be able to model processes that embrace different paradigms. To demonstrate such property, in Fig. 2.2 we present an instance of the metamodel describing the agile process Scrum [2,8].

We choose an agile development process to show the flexibility of our approach; in fact, agile processes, and in particular Scrum, are intrinsically unpredictable, although a control mechanism is used to guarantee flexibility, responsiveness, and reliability of the results. These characteristics could make difficult the implementation of such a rigorous measurement framework. Thanks to the independence between process and measurement module, our metamodel could seamlessly superimposes a measurement framework to agile-based projects.

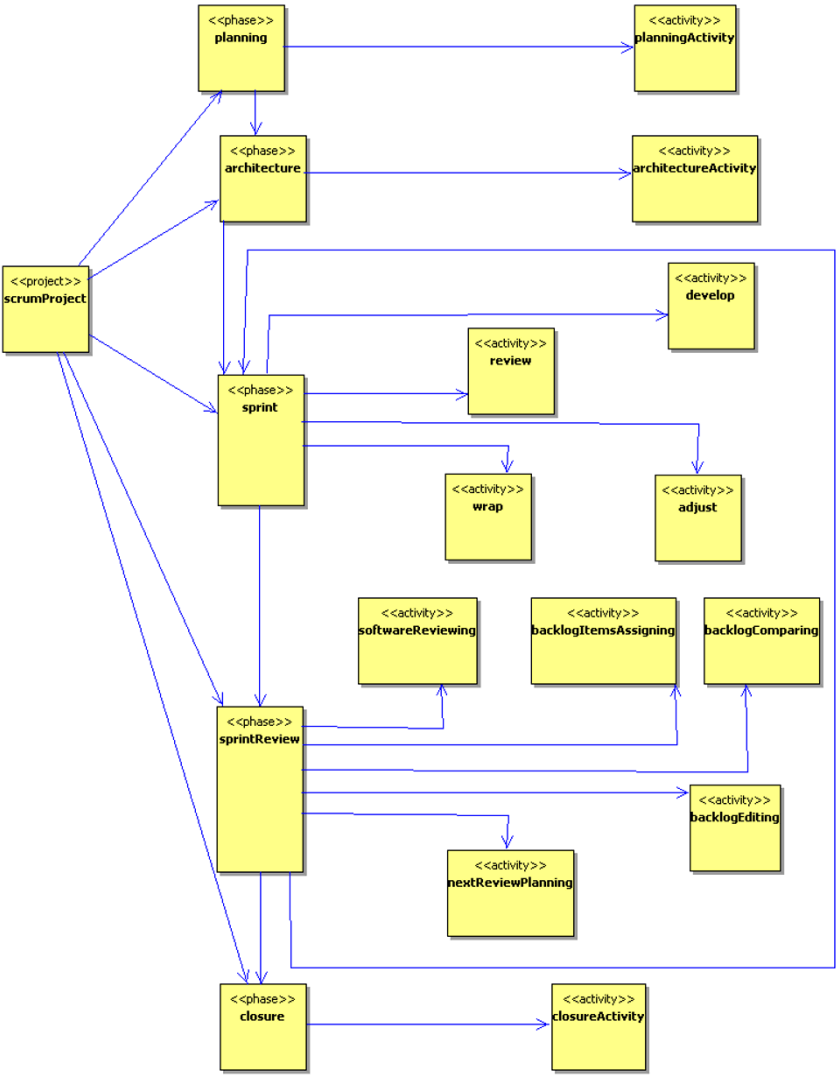

Fig. 2.2 Model of the agile development process Scrum.

3. PMLite: Process Monitoring Lite

The requirements that have driven the development of PMLite are essentially three.

First of all, we wanted to develop a web-based application that fully adopts and verifies our open metamodel representation, allowing managers to model any type of process and organize measurement campaigns to gather all needed attributes.

Secondly, we wanted PMLite to be essentially an easy-to-use tool, with a gentle learning curve, that could be adopted also in small software houses and open source development groups, without any particular installation effort. For this reason, we choose to propose a data collecting technique based on surveys instead of automatic probes, lowering installation problems and development effort. Furthermore, the huge number of applications used during software development and during support activities makes difficult to implement automatic probes that will extract measurable attribute from a suitable set of applications work products.

Finally, PMLite is a first step toward developing a complete process monitoring platform, which could exploit our metamodel approach for generic monitoring generic business processes.

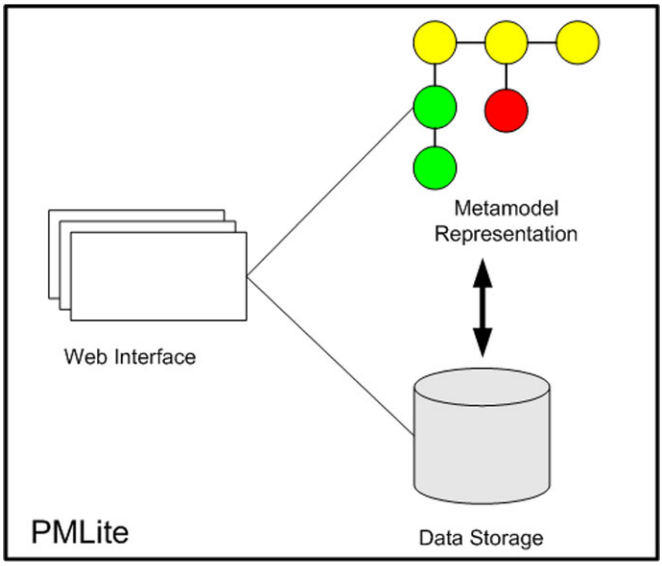

Fig. 2.1 Conceptual structure of PMLite.

3.1. PMLite Description

The conceptual structure of PMLite is depicted in Fig. 2.1. Both web pages and data storage have been designed basing on the metamodel structure and classes, and the supplied activities start from defining, for any project, activities and phases of the relative development process, the measurable attributes to be retrieved, and the questions that compose the surveys.

Fig. 3.2 PMLite homepage.

Fig. 3.3 Interface for the insertion of a new activity.

To better describe the structure of the tool, we concentrate on three key actions: *i)* definition of the process, *ii)* definition of the survey, and *iii)* execution of the survey.

Definition of the process.

The homepage of PMLite (Fig. 3.2) allows the access to specific functions of the application.

The first step is the definition of the specific development processes in terms of phases, activities (see Fig. 3.3), and relationships between phases and related activities. Further, the tool allows to define the transitions between the activities them selves. In this way, the process is well defined and projects can be linked to the specific process.

Then, managers have to define the attributes over which the analysis will be based. The tool makes simple the insertion of measurable attributes (see Fig. 3.4) but the procedure of specification of them is critical, since they will be the basis for the

definition of questions, of surveys, and, consequently, of the whole measurement framework. At the time being, an attribute is only characterized by its name and description.

Definition of the survey.

In its full implementation, the metamodel requires attributes to be retrieved by automatic extractors (i.e. instances of trigger classes). However, for the reasons explained above, PMLite simulates the automatic triggers via specific question sets; users interact with PMLite by answering to the questions associated to the current activity.

PMLite gives specific interfaces to fulfil these actions. In particular, the interface in Fig. 3.5 presents a complete set of questions. Each question is characterized by a text and three possible types of answers (*clear text*, *single choice*, and *multiple choices*). Each question is then gathered in a specific *Question Set*, which, in turn, is associated to a specific process phase or activity. This allows the system presenting to developers the questions sets concerning the specific development action they are performing.

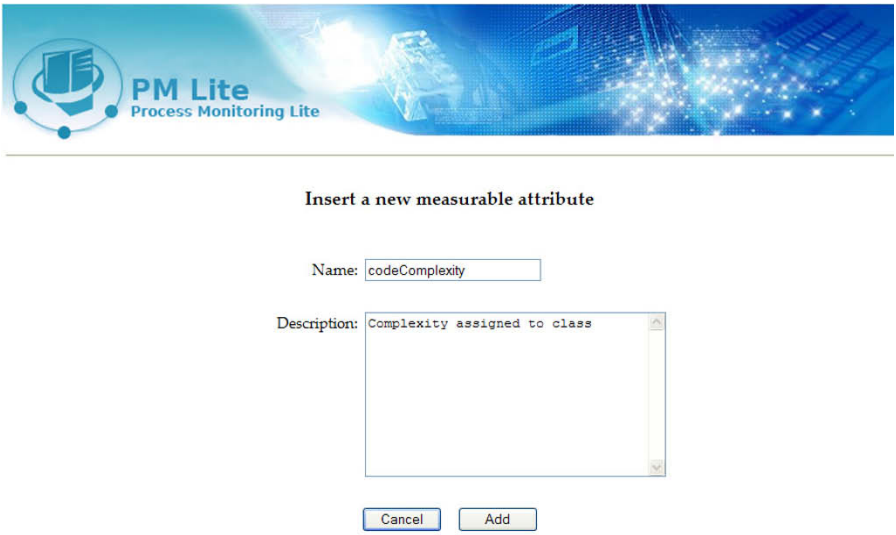

Fig. 3.4 Interface for the insertion of a new attribute.

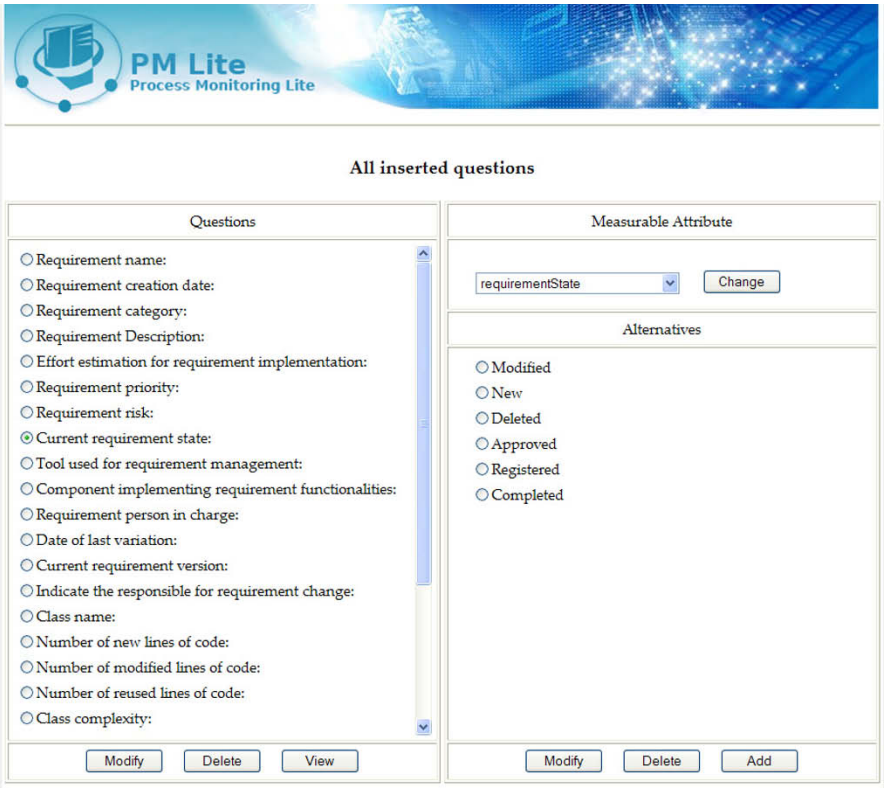

Fig. 3.5 Interface for the management of questions.

Execution of the survey.

As said above, each questions set is relative to a specific phase or activity of a process, therefore it is important that the tool will presents to users the questions that are specific of the activity or the phase they are working on. In the interface in Fig. 3.6 first, developers choose the current activity or phase, then, PMLite presents to them the relative series of questions.

The approach followed in developing PMLite, at a first sight, could seem too intrusive, since developers have to manually interact with the survey interface any time they start a new activity or a new process. However, this allows adopting PMLite even in lightweight development environments where no configuration management or event tracking is available.

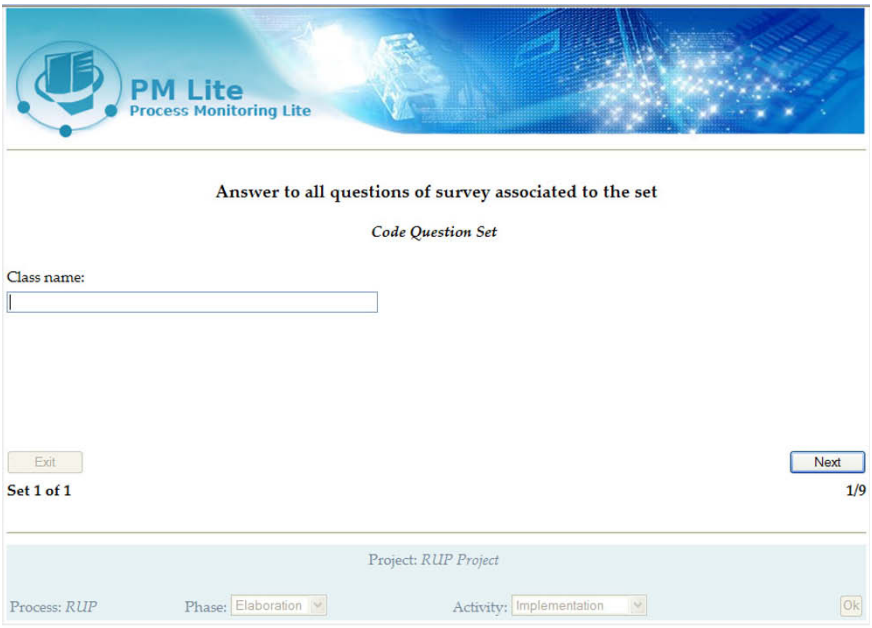

Fig. 3.6 Interface for the execution of the surveys.

4. Future Works

PMLite is only the first step in the development of a complete and automatic process monitoring environment, which will be fully transparent and non-intrusive for the developers, but allows us to test and proof our approach.

Our metamodel has been fully exploited for the designing of the structure of a more complete monitoring open source tool, Spago4Q [9].

We plan to exploit PMLite also for the definition and as proof-of-concept of specialized GQM, as for instance for the quality assessment of Open Source products and for the complexity evaluation of business process.

5. Conclusions

In this paper we presented our new open source tool, PMLite, which implements our study [2] of a metamodel to completely formalize an enterprise-level process monitoring framework. PMLite is directed to these organizations that manage projects adopting different development processes and want to have a snapshot of the global status of the current works. The methodology proposed, although could seems intrusive for developers, has the unique strength of being adaptable not only for development process monitoring, but also for generic process representations.

Acknowledgments

This work was supported in part by the European Union within the SecureSCM project in the FP7-ICT Programme under contract n. AO213531, and by contract/grant sponsor FIRB research fund of MIUR, research project TEKNE (contract/grant n.RBNE05FKZ2).

References

[1] Basili VR (1992) Software modeling and measurement: The goal question metric paradigm. In MD University of Maryland, College Park, editor, Computer Science Technical Report Series, CS-TR-2956 (UMIACSTR-92-96)

[2] Bellettini C, Colombo A, Damiani E, Frati F (2007) A metamodel for modeling and measuring scrum development process. In Springer Berlin, editor, Agile Processes in Software Engineering and Extreme Programming - Lecture Notes in Computer Science, 4536:74-83

[3] OMG Group (2008) Mof - metaobject facility. www.omg.org/mof/. Accessed January 2008.

[4] OMG Group (2008) Spem software process engineering metamodel. www.omg.org/technology/documents/formal/spem.htm. Accessed January 2008

[5] Ventura Martins P, Rodrigues da Silva A (2005) Pit-p2m: Projectit process and project metamodel. In Proc. of OTM Workshops, Cyprus, 516-525

[6] PMLite (2007) Process monitoring lite. sourceforge.net/projects/ pm-lite/. Accessed January 2008.

[7] F. Ruìz, A. Vizcaìno, F. Garcìa, and M. Piattini (2003) Using xmi and mof for representation and interchange of software processes. In Proc. of 14th International Workshop on Database and Expert Systems Applications (DEXA'03), Prague, Czech Republic

[8] K. Schwaber (1995) Scrum development process. In Proc. of OOPSLA'95 Workshop on Business Object Design and Implementation, Austin, TX, US

[9] Spago Solutions (2007) Spago4q. www.spago.org/. Accessed January 2008