PEM: Experience Management Tool for Software Companies

Emanuele Danovaro
Free University of Bozen-Bolzano
Emanuele.Danovaro@unibz.it

Tadas Remencius
Free University of Bozen-Bolzano
Tadas.Remencius@unibz.it

Alberto Sillitti
Free University of Bozen-Bolzano
Alberto.Sillitti@unibz.it

Giancarlo Succi
Free University of Bozen-Bolzano
Giancarlo.Succi@unibz.it

Abstract

Process control and improvement are keys to successful businesses. A working Experience Factory helps to achieve them but it is not easy to implement. The PROM Experience Manager (PEM) is designed to facilitate such implementation with a flexible visual interface to an experience base populated of metrics collected non invasively.

Categories and Subject Descriptors D.2.9 [**Software Engineering**]: Management; K.6.3 [**Computing Milieux**]: Software Management;

General Terms Management, Measurement.

Keywords Experience Manager; Dashboard; Metric Interpretation.

1. Introduction

Effective management of the processes of the company has been recognized as one of the keys to success in business and has been the focus of a number of standards, such as ISO 9000, ISO 15939, TQM, CMMI, 6 Sigma, etc. However, it is not easy to achieve in practice.

Managing the processes effectively means that managers and developers need to be able to understand what and why is happening, to be able to affect (e.g., change, improve) the processes, to evaluate their results, and to learn from them (Figure 1).

The Experience Factory (EF) [1] is an infrastructure that can facilitate this. It is designed to capture the experiences and products of the life-cycle, to package and to prepare them for later reuse. The success of the EF and the benefit it can bring to a company is largely dependent on the acceptance of the framework by the employees and on their willingness to participate in it [2].

PROM Experience Manager (PEM) is a tool designed **(a)** to provide a framework for automatic data collection using the PROM infrastructure [3] and **(b)** to help motivate users to participate in the forming and use of the EF by providing a visual interface to the experience base of the company. The backbone of PEM is a goal oriented top-down approach – we use an adaptation of the GQM.

The GQM (Goal/Question/Metric) [4] is a measurement model composed of three levels: (a) conceptual (goals), (b) operational (questions defining the goals in the quantifiable way), and (c) quantitative (set of metrics associated with questions to answer them in a measurable way).

In addition to the GQM, we also let users create and share personalized collections of views not tied directly to the underlying model. For example, alongside the visualization of a GQM metric, one might place a view containing tasks related to a goal or with user comments about usefulness of particular metric. In this way the tool creates a bottom-up feedback layer that facilitates user collaboration and helps to discover personalized benefits from the experience.

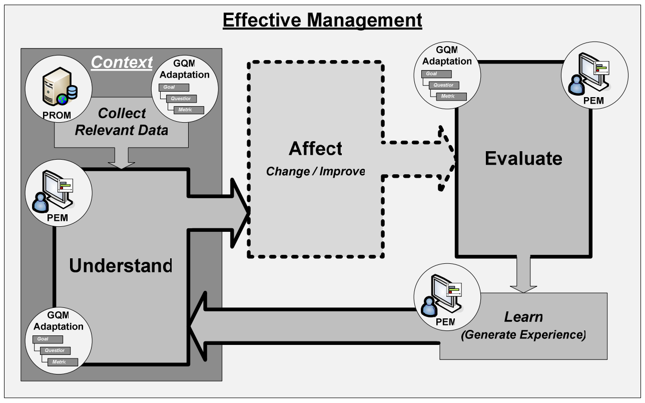

Figure 1. The role of PROM Experience Manager (PEM).

2. Architecture of the Approach

We adapt the GQM by extending it with optional goal properties and introducing a new "abstraction" level (Figure 2) to "interpret" the values of the metrics.

We also support a hierarchy of goals, from higher level (business) to low-level measurement goals, as in the GQM$^+$ Strategies approach [5]. The new goal properties define:

- **Expiration**: when and if the goal expires (deadline);
- **Condition**: formula defining if the goal has been reached (possible values: *reached*, *not reached* or *failed*).
- **Prediction / Estimation**: formula predicting if the goal is reached (possible values: *reached* or *failed*).

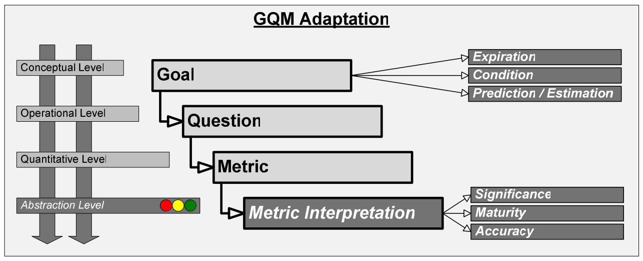

Figure 2. Adaptation of the GQM approach.

The abstraction level is composed of metric interpretations that take the values of the metrics and convert them to an abstract uniform scale of "goodness".

Different metaphors can potentially be used to construct the scale of abstraction. We use the metaphors of the traffic light, which can assume three values: (a) green (OK; everything is fine); (b) yellow (warning; neither good nor bad and can potentially result in a problem); and (c) red (problem; attention required). The metaphor of the traffic light is easy to understand at-a-glance and simplifies the interpretation of the metrics also for non IT/experts.

Metric interpretations have several optional properties that are used to help evaluate interpretations and metrics:

- **Significance**: its importance (influence, weight) to the goal on a scale from 0 (no effect) to 100 (perfectly aligned with the goal).
- **Maturity**: the level of trustworthiness of the interpretation. It can have one of three values: unproven (default), proven, and unreliable.
- **Accuracy**: how often the metric represents the goal correctly on a scale from 0 (never) to 100 (always).

3. Structure of the Solution

PEM is composed of two logical parts: (1) an Ajax-based web-based system that uses Apache Tomcat (web server) and PostgresSQL (DBMS); (2) a server side Java application, for metric storage and external data import.

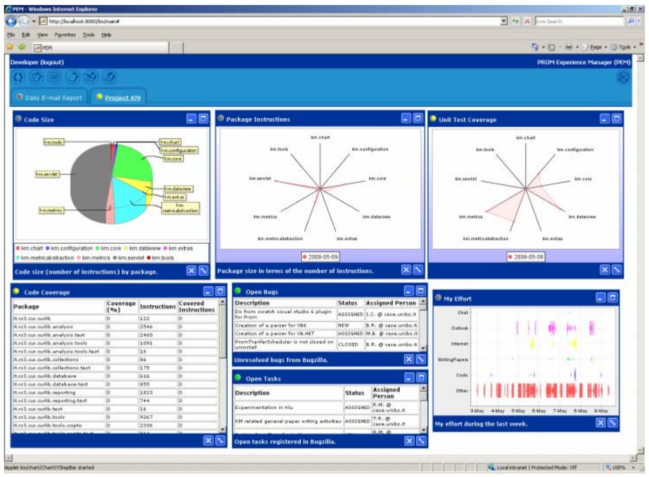
Developer (logout)
PROM Experience Manager (PEM)
Daily E-mail Report
Project KM
Code Size
Package Instructions
Unit Test Coverage
Code size (number of instructions) by package.
Package size in terms of the number of instructions.
Code Coverage
Open Bugs
Unresolved bugs from Bugzilla.
Open Tasks
Open tasks registered in Bugzilla.
My Effort
My effort during the last week.

Figure 3. Screenshot of the system.

PEM uses the dashboard-based approach for its GUI (Figure 3), which is composed of three main elements: (1) toolbar with control buttons; (2) tab-based navigation between available perspectives; and (3) views contained inside the active perspective.

Views are of different types based on what their function is. However, all views have the same visual structure: (a) title bar with metric interpretation "light bulb", view name and state control buttons (minimize / maximize / restore); (b) view contents (e.g. chart, data, text, GQM, etc.); and (c) status bar with view description and additional controls dependent on the user rights (e.g., resizing, removal).

4. Experience

Our experience from deploying the tool in software companies showed us that PEM makes managers and developers understand the value of a metrics program. They can appreciate visually that metrics exist (really!) and evolve in time without any additional effort for collection, providing useful information on the development process and product. This overcomes a lot of existing prejudices on metrics programs.

Moreover, using metric interpretations motivates managers and developers to think about them and thus to understand the metrics better. The challenge becomes then to come up with a good interpretation formula – which is in a sense good, as it requires a more consistent understanding of the metrics within an organization. A more consistent interpretation of metrics is further promoted by the visualization of the interpretations.

We are now starting a formal review process with ~20 industrial developers; the results will be ready in the next year.

References

[1] V. Basili, G. Caldiera and D. Rombach, "Experience Factory", Encyclopedia of Software Engineering, Wiley, Vol. 1, 476-496, 1994.

[2] C.B. Seaman, M.G. Mendonca, V. Basili and Y.M. Kim, "User Interface Evaluation and Empirically-Based Evolution of a Prototype Experience Management Tool.", Transactions on Software Engineering, Vol. 29, Issue 9, 838-850, 2003.

[3] A. Sillitti, A. Janes, G. Succi, and T. Vernazza, "Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data", Proceedings of the 29th Conference on EUROMICRO, 336, 2003.

[4] V. Basili, G. Caldiera and D. Rombach, "The Goal Question Metrics Approach", Encyclopedia of Software Engineering, Wiley, Vol. 1, 528–532, 1994.

[5] V. Basili, J. Heidrich, M. Lindvall, J. Munch, M. Regardie and A. Trendowicz, "GQM$^+$ Strategies - Aligning Business Strategies with Software Measurement", Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM), 488-490, 2007.