



Project DEPLOY
Grant Agreement 214158
*“Industrial deployment of advanced system engineering methods for high
productivity and dependability”*



DEPLOY Deliverable

D10 D11.2 Initial Data Collection Framework

Public Document

31 January 2009 – V1.0

<http://www.deploy-project.eu>

Executive summary

This deliverable outlines the data collection framework to be delivered to the industrial partners for ensuring the systematic and consistent collection of measurements.

This deliverable presents data collection requirements for a simple and automated tool. Based on recommendation for measurements, an appropriate architecture will be developed.

An initial version as a simple implementation based on spreadsheets is described. This system could evolve later in a more elaborated (web-based) system. A set of relevant tools for supporting this task are also reviewed.

Finally the requirements and design of a tool for measuring Event-B related products (Event-B specification and proofs) and processes (formalizing, proving, documenting...) are also described.

The scope of this deliverable is the measurement infrastructure. The methodological aspects of measurement are covered in the complementary deliverable: D7, Measurement Methodology Guide

History

Date	Version	Author	Description
23/10/2008	0.1	C. Ponsard, S. Saadaoui	Deliverable structure
12/11/2008	0.2	S. Saadaoui	Requirements for data collection framework
15/12/2008	0.3	C. Ponsard S. Saadaoui	General data collection architecture and plug-in requirements
2/1/2009	0.4	C. Ponsard	Background section
5/1/2009	0.5	C. Ponsard	Architecture refinement.
9/1/2009	0.6	C. Ponsard	Consolidation.
30/1/2009	0.7	Marta Płaška C. Ponsard	Corrections after review
31/1/2009	1.0	C. Ponsard	Final corrections

Table of contents

Executive summary.....	2
1 Introduction	4
1.1 Goals of the measurements workpackage and of the deliverable	4
1.2 Scope of this deliverable	4
1.3 Structure of this deliverable	4
2 Background on measurement infrastructures	5
2.1 Principles.....	5
2.2 Remote Measurements Collection	6
2.3 Classification of existing tools	6
3 Requirements for the measurement infrastructure.....	8
4 Architecture of the measurement framework	9
4.1 Complete measurement infrastructure	9
4.2 Initial architecture	10
5 Specification of RODIN measurement plug-in	12
6 References	15

1 Introduction

1.1 Goals of the measurements workpackage and of the deliverable

Formal methods have been applied quite successfully in a number of industrial case studies and are even adopted in specific industrial fields. However, they are still not widely used in the commercial software development, even in safety/security/business critical sectors such as automotive, aerospace, e-business. Over time reasons for this have been widely studied, a number of obstacles and often misconceptions (“myths”) have been identified [Hall90][Bow95] such as the difficulty of the mathematics, the incompatibility to use them with customers, the effect on the costs and delay, the total incompatibility with traditional development methods, the lack of support and tools, etc. Guidelines (stated as “commandments”) have also been suggested [Bow95][Bow06] to address those.

For adoption, especially at the management level, there is a necessity to provide numeric evidence that formal methods are applicable and have positive influence on the development process and on the overall quality of the final product. Formal methods can effectively improve safety and reliability; still organizations do not always assign direct value to improvements in those areas and therefore, in the absence of measurements, they are perceived as additional cost [St03]. Management also needs new progress indicators as the introduction of formal methods requires more effort in the early development stages while typically reducing costs at the coding and testing phases (e.g. in B development, unit testing can be completely suppressed). In the absence of such indicators, the management will interpret a project as not progressing.

This situation heralds the need for deploying appropriate measurement strategy in conjunction with the deployment of formal methods. The goal of this deliverable is to propose an adequate measurement infrastructure for collecting the data in the context of the measurement methodology described in D7. This deliverable presents a complete infrastructure. However, initially only a subset of this infrastructure is considered; it will be extended subsequently. This stepwise approach is recommended to ensure the tools are adapted and that the maturity of the corresponding development process is matching the pace [Eb05].

1.2 Scope of this deliverable

The scope of this deliverable is the measurement infrastructure. The methodological aspects of measurement are covered in D7 - measurement methodology.

1.3 Structure of this deliverable

This deliverable is organised as follows:

- Section 2 gives background of measurement infrastructure. It presents literature review of related principles and classification of tools
- Section 3 gives the requirements of the DEPLOY measurement infrastructure
- Section 4 details the architecture at two levels: a complete, ideal architecture and a simplified architecture developed as a first step to avoid deploying heavy procedures from the beginning.
- Section 5 details the specifications of the components to be eventually developed in the project leading to its comprehensive architecture.

2 Background on measurement infrastructures

2.1 Principles

The need for measurement tools is as obvious as the tool support in any other discipline of software engineering. [Eb05] and [Eb07] provide guidelines for measurement tool applications. The reasoning and goals of the guidelines provided by the above references are summarised below:

- The efficiency of the software measurement process depends on the level of automation of the tools. One should try to achieve a complete tool-based solution for measurement process. However, requirements for the tools have to be carefully worked out. Moreover, implementing the tool needs to be planned according to the measurements needed.
- Measurement tools should cover the whole software measurement process, starting with establishing measurement, and continuing with planning, performing the measurement and exploring the results for process, product and resource evaluation. This helps in embedding the measurement into software development environments for assessing, improving and controlling the original IT processes.
- The philosophy of the measurements tools should be applied in the proper order. The applied tools should be embedded in a compatible software measurement framework (as described in D7 - Measurement Methodology Guide)
- Specific parameters of the software development environment should be known so as to ensure correct and complete input information for the tool. A thorough analysis of the empirical aspects such as effort and cost is an imperative precondition for the proper use of a selected tool (for the right use of the right measurement tool).

2.2 Remote Measurements Collection

Data collected through the tool should eventually be stored in a central location. Internet can be used for the data transfer across organization boundaries. Software e-measurement means the application of the World-Wide Web in order to support software measurement processes, activities and communities. E-measurement can be considered for:

- Transferring raw data versus transferring computed metrics, depending on the presence of a local facility.
- Ensuring an external storage of the data in the absence of a measurement infrastructure.
- Providing presentation of the measurement in a meaningful way

In all these cases, security guarantees have to be provided about the confidentiality of the data gathered and to preserve their integrity throughout the data lifecycle.

2.3 Classification of existing tools

Two classifications have been found in the literature. The first, given in [Eb07], is mainly structured around three kinds of measurement tools: process, products and resource, followed by more specific tool for presentation and training. Here is a summary of this classification:

- **Process Measurement and Evaluation**, helpful for application during the measurement of the software process phases, components and activities.
- **Product Measurement and Evaluation**. Those are addressing various stages of the development process: requirements engineering, software design, program evaluation, software testing, software maintenance
- **Resource Measurement and Evaluation**. The following dimensions of software resources components and activities are supported: productivity, performance and usability.
- **Measurement Presentation and Analysis**, with Excel or similar spreadsheet software as the most widespread systems. Specific reporting systems can be used for the presentation such as crystal reports, BIRT, JaspertReport, etc.
- **Measurement training**: specific tools helpful for learning and understanding the phases, components and activities of software measurement including their theoretical background.

A second classification was proposed by [Fen97] and is structured following a data collection process with planning, forecasting, collection, analysis and storage steps and related tools. This classification can be summarised as follows:

- **Planning tools**, to help in planning the overall program, including deriving questions and metrics from the goals. [Lav00] details some tools for providing automated support for the QGM measurement process.
- **Forecasting tools** help forecast likely effort, schedule, or defect information. There are many cost estimation packages, implementing function points counting, public models such as COCOMO or proprietary models such as SLIM. Reliability modelling tools help to track defect information, and test support tools use defect discovery histories to forecast when testing will be complete.
- **Data collection tools**: static or dynamic can be used to measure size and structure of code, or more recently of models. Moreover, they can be used to diagnose latent errors, not discovered using other techniques. Other tools can compare files, track change histories; count requirements and design components, etc.
- **Data analysis tools**. Most of the analysis is performed using simple tools like spreadsheets and statistical tools. The spreadsheet nicely combines analysis and reporting capabilities using both tables and diagrams. Statistical tools can generate classification trees, draw box plots, and perform standard analysis to test hypotheses. Additionally more specialized tools can be used in more specific context for example to computing control flow path, identify test coverage.
- **Data storage tools**. A repository is mandatory for any measurement infrastructure. Data is best stored in a conventional database or as an extension to a configuration management or project management system. A database system can provide a higher degree of confidentiality and can also support data of several projects to permit an organizational or corporate view of project results and trends.

Note that all of these tools are directed primarily at measurements (e.g. databases, spreadsheets).

In the scope of our measurement infrastructure, we will develop an architecture covering both process and product measurements. The main tools will be data collection and data analysis tools.

3 Requirements for the measurement infrastructure

Before detailing the architecture of our proposed infrastructure, the main underlying requirements are specified. Those requirements were collected through the interactions with the industrial partners during the industrial kick-off meetings. Some suggestions about the automatic data collection were recommended by the academic partners.

Table 1 presents the requirements identified for our measurement infrastructure.

Id.	Name	Description	Link
R-01	Simplicity	The measurement infrastructure should be simple enough to minimize disruption to normal work patterns.	
R-02	Automation	The measurement infrastructure should be automated to the maximum extent, contributing to simplicity. The measurement process is nevertheless not required to be fully automated as some parts cannot be automated.	R-01
R-03	Integration	The measurement infrastructure should be integrated with other tools supporting the development process where it can find useful data. Typical tools to be considered are: timesheet system, versioning system, integrated development environment of analysts and developers, test management system, defect tracking system.	R-01, R-02
R-04	User-friendliness	The manual part of the data collection should rely on clear, complete and precise forms.	R-01
R-05	Database	The measurements should be stored in a central database.	
R-06	Reporting	A reporting facility should be available to query information in the database and present it in an understandable form (tabular or graphical)	
R-07	Authorisation	In the case of e-measurements, data transferred outside the enterprise boundaries should be authorised on the basis of agreed metrics (see D7)	
R-08	Confidentiality	In the case of e-measurement, confidentiality of the transferred data should be preserved by the hosting infrastructure.	

Table 1. Requirements for the measurement infrastructure

4 Architecture of the measurement framework

This section details our proposed measurement framework. It first describes a complete, ideal architecture; and then presents its simplified version. The simplified version fulfils the initial data collection needs, mainly for the requirements & specification phase, without considering integration with the industrial tool chain and development environment. Both architectures are based on the RODIN platform instrumented with a specific measurement plug-in.

4.1 Complete measurement infrastructure

The proposed architecture of the measurement infrastructure is shown in the Figure 1.

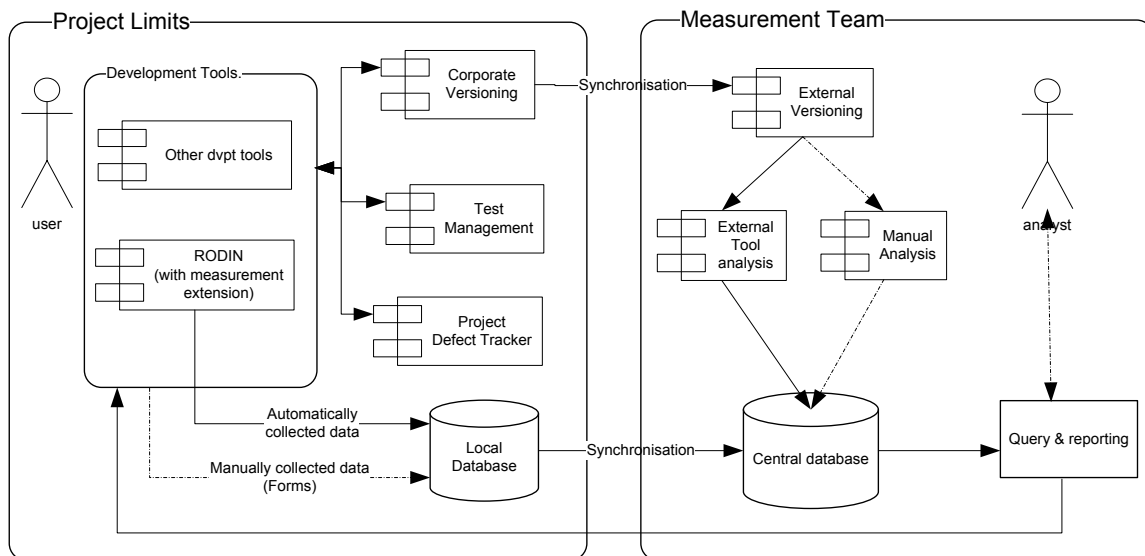


Figure 1. Complete measurement infrastructure.

The measurement infrastructure is divided into two parts: the project part and the part dedicated to measurement activities. This division can support an e-measurement scenario, where the measurement team is not located in the industrial partners' site, but in an external organization (as in DEPLOY), but also a measurement process remains strictly internal to the company, both parts are then located in the same project. There can be multiple instances of the left (project) part but only one instance of the right (measurement) part.

The overall idea is that the impact of the measurement has to be kept minimal on the project side: the installation should be easy and maximum information is automatically collected through interfacing with the industrial partners' tools. The manual part is restricted to a short number of forms to be filled in either periodically (e.g. for timesheet

if the corporate system is not used), or at given project milestones (e.g. completion of some deliverable). Note that some metrics will be collected on the project site, especially for those useful in providing a direct feedback to the analyst/developer (e.g. Event-B specification metrics in the RODIN platform) while others will be computed less reactively, on the measurement side (e.g. data for the project manager).

Most of the burden is borne by the measurement team that has to collect relevant information using the collection means in place and analyze them using the most appropriate tools (see chapter 1) or even manually if there is no other way. All the collected information is stored in a central measurement database which also serves as a project memory. The synchronization is ensured between a local and central database. Moreover, it may also be necessary to get direct access to some development products; therefore, another synchronization process replicates them between the corporate versioning system and a measurement versioning system. Note that:

- in the scenario of a local deployment, direct access to the corporate versioning could be provided: there is no need to replicate those products.
- in the scenario of external measurement, synchronization must be authorized.

4.2 Initial architecture

The initial architecture is a simplified subset of the complete architecture presented in the section 4.1. It is deployed as a first step because the complete architecture:

- is not necessarily deployed right away: it can be deployed progressively as the deployment is progressing through the development lifecycle.
- would not be adopted if deployed in one step: an incremental approach is wiser: introducing first simple tools, let people adopt them, see what works, what is useful, what needs to be tuned, what is missing.
- cannot be deployed at the moment, because some tools are still under development; for example: the RODIN versioning, the connection with a requirement tool, the measurement plug-in.

This simplified subset is nonetheless functional and thus supports the expected measurement activities. Those requirements are explained in the section 4.2.1 followed by the detailed simplified architecture in the section 4.2.2.

4.2.1 Requirements for the initial architecture

The simplified architecture should support the measurement plan for the first phase of the pilot study. This will consist of the following documents and models:

- Software requirements specification
- Problem frames (optionally, for Bosch)
- Event-B models annotated with links to requirements

- Semi-formal design documents, mixing text and some notation such as UML diagrams, finite state machines, tabular notations.

The product metrics will be computed based on the processing of the above products.

In addition, the process metrics typically require the monitoring of the following processes:

- requirements engineering activities: writing, structuring, correcting requirements, optionally problem frame modelling
- Event-B modelling activities: formalizing in Event-B, proving, model correction, and model validation (using the RODIN/PROB animator).

Monitoring these activities can be partly automated through the availability of a RODIN monitoring plug-in (see chapter 5). For other matters, it is necessary to rely on the user to keep track of:

- Effort spend, through some time sheet based on the identified activities (if a corporate time sheet is available, it can be used with corresponding activity code)
- Change log between successive versions over time (measured at completion of milestone and at intermediary steps of partial completion). A clear distinction will be made between different types of changes in order to be able to analyse their impact. The following classification will be developed:
 - defects in requirements: missing, ambiguous, conflicting or over specified requirements
 - change request in requirements: addition, deletion, precision, etc.
 - corrections in Event-B model: more requirements covered, restructuring (through refinement), correction (proof fixed), etc.

To cover this, it is necessary to keep track of:

- Full document and model snapshot at the measurement time. In the absence of internal versioning, versions have to be exported to the measurement system.
- Effort log between snapshots
- Change logs between snapshots

They will be collected in the most automated way, based on the capacity of the measurement plug-in to export the captured data and monitor the activity of the analyst. The export operation will require authorisation. In the event a corporate firewall is blocking the export, a zip file will be produced and can be mailed or posted manually instead, possibly after inspection.

4.2.2 Description of the initial architecture

The initial architecture is sketched in Figure 2. It assumes a deployment where the measurement activities are outsourced. The simplification is mainly on the project side:

- The sharing of products to be measured is done through the DEPLOY shared file system (BSCW). To cope with the lack of versioning in RODIN, the Event-B projects are zipped and tagged with dates at the agreed measurements milestones.
- The local database will be directly managed by the RODIN measurement plug-in.
- Forms will be implemented as spreadsheet files sent by email or shared through the BSCW.

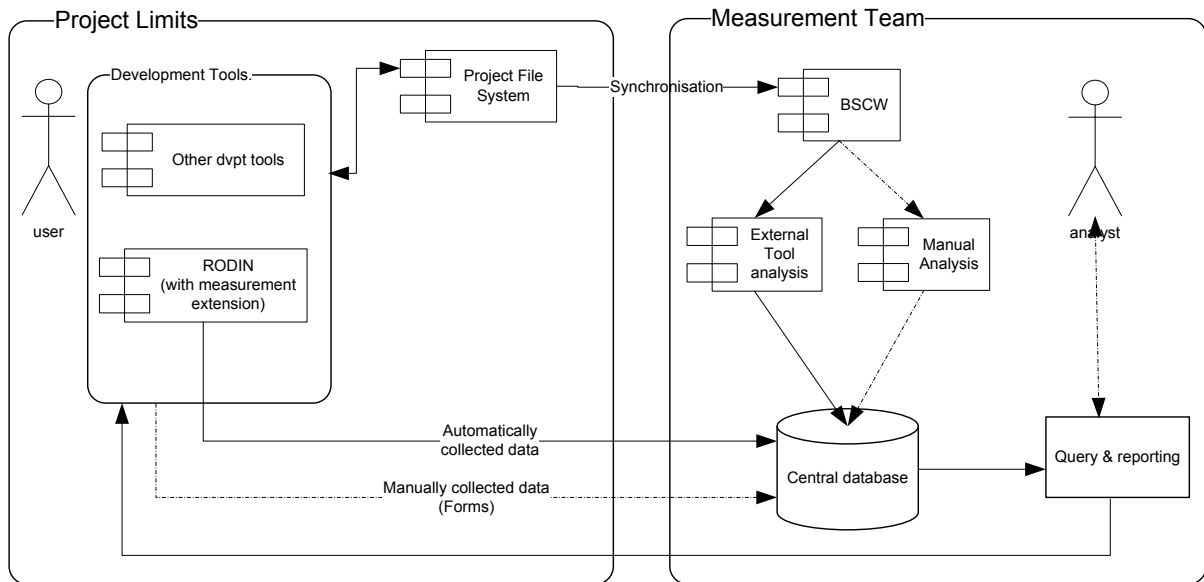


Figure 2. Initial Measurement Infrastructure.

5 Specification of RODIN measurement plug-in

The main component to be developed for the initial architecture is the RODIN measurement plug-in. It will provide information both about the model itself and about the process of building the model. It has a double purpose:

- provide feedback to the user about the quality of the Event-B model being built and about potential problems in it or in the way it is being built.
- automate the data collection process for the measurement and assessment WP. This data collected will be analyzed to identify global transfer (increase in model quality, size, complexity, etc), tool shortcomings (usability, prover), modelling issues (to be addressed by training, language, tool evolution, etc), etc.

This work is planned for project year 2. This chapter introduces the requirements and the general architecture of the tool.

5.1.1 Requirements

The plug-in will fulfil the following functional requirements

- evaluation of low-level model metrics, such as number of events, refinements, etc
- evaluation of relevant quality metrics such as complexity, maintainability, etc
- evaluation of tasks related metrics such as time spend in modelling, proving or other activities (possibly based on other plug-ins) such as requirements, model-checking.
- the user will be given the ability to enable or disable the collection of task related metrics.

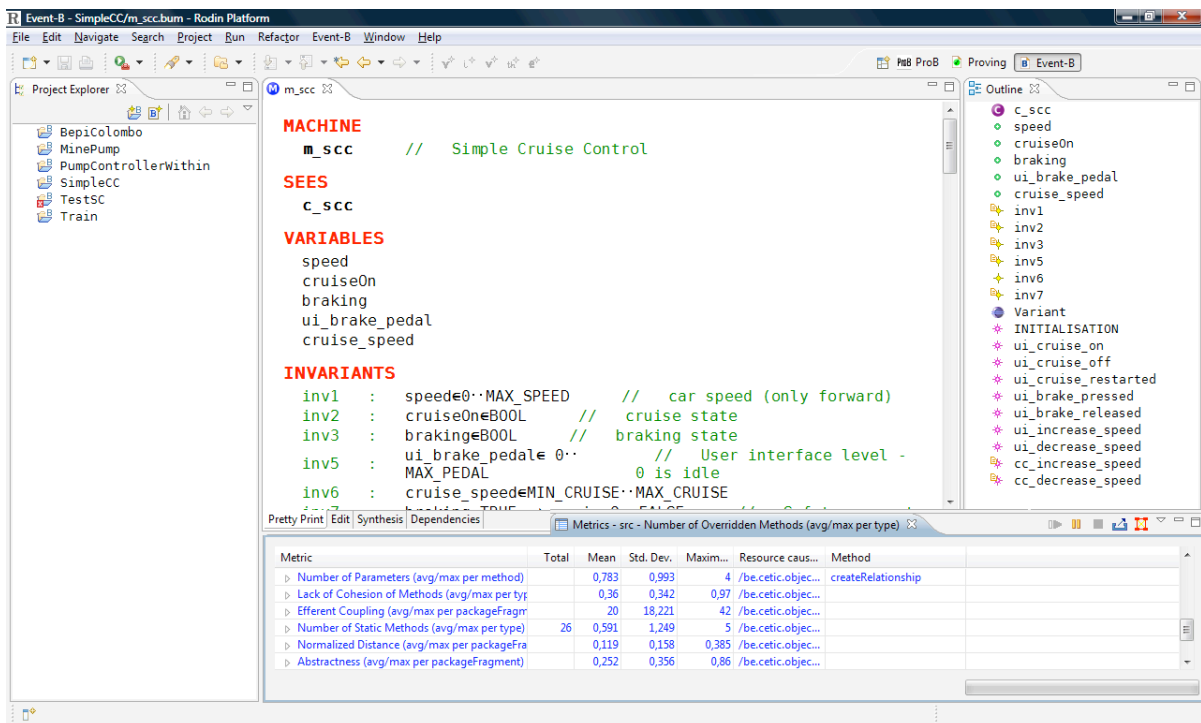


Figure 3. Mock-up of the metric plug-in.

The following non-functional requirements are also identified:

- usability: integration into RODIN, in the Event-B perspective as an additional view. The minimal interface will show the value of the metrics selected for the currently loaded model or model element. Extended interface could show under graphical form and evolution over time. Figure 3 shows a mock-up of the graphical integration of the metric plug-in as a specific view displayed at the bottom of the Event-B perspective.

- reactivity: metrics will be updated regularly (at an adequate pace, possibly incrementally) or at user request
- efficiency: metric evaluation will not significantly slow down the interactivity of the RODIN platform
- security: strict data management policy to enforce confidentiality of the models, especially the user must be able to see if the plug-in is enabled and no data can leave the tool without the user consent

5.1.2 Architecture of the solution

The implementation will be carried out as plug-in to the RODIN platform on Eclipse. It will be composed of the following modules.

- **RODIN interface module.** It provides access to the model through the RODIN API in order to extract the useful information from each metric.
- **Eclipse interface module** Additional access to gather other Eclipse information (such as the currently selected perspective, user activity) can also be required for process monitoring. Third party plug-in such as Mylyn [Myl] could also be useful for this purpose, as it monitors work activities to identify relevant information (making tasks context explicit).
- **Metric computation module.** The module will use the RODIN interface module to query the specification and collect information to compute metrics.
- **Reporting module.** This module will provide a local consultation of the collected data and computed metric.
- **Export module.** It will export the collected metrics to the central database. This export can be set to take place automatically with the prior user consent.
- **Configuration module.** This module will allow the user to select the computed metrics and the exporting mode.

6 References

- [Eb05] Ebert Christof, Dumke Reiner, Bundschuh Manfred, Schmietendorf Andreas, Best Practices in Software Measurement, Springer, 2005.
- [Eb07] Ebert Christof, Dumke Reiner, Software Measurement: Establish, Extract, Evaluate, Execute, Springer, 2007
- [Fen97] Fenton, N. E., and Pfleeger, S. L. Software Metrics – A Rigorous and Practical Approach, 2nd Edition, PWA Publishing Company, Boston, MA, 1997.
- [Lav00] Lavazza, L. 2000. Providing Automated Support for the GQM Measurement Process. *IEEE Softw.* 17, 3 (May. 2000), 56-62.
- [Myl] The Eclipse Mylyn project, <http://www.eclipse.org/mylyn>