The use of a meta-model to support multi-project process measurement

Alberto Colombo1, Ernesto Damiani1, Fulvio Frati1, Sergio Oltolina2, Karl Reed3, and Gabriele Ruffatti2

1Department of Information Technology - University of Milan - Italy
{colombo, damiani, frati}@dti.unimi.it
2Engineering Ingegneria Informatica – Italy
{sergio.oltolina, gabriele.ruffatti}@eng.it
3Department of Computer Science and Computer Engineering – La Trobe University - Australia
kreed@cs.latrobe.edu.au

Abstract

In today’s environment, software companies are engaged in multiple projects delivered on heterogeneous platforms for a wide class of applications in disparate application domains. They are increasingly engaged in the co-development of software systems through joint software development projects including staff from partners and customers as well as their own. As a result, they must support multiple software development processes while trying to guarantee uniform levels of process enactment, and product quality across all projects. Our approach is capable of providing process measurement in a joint-project, multi-process model business environment. It is based on a simple meta-model for computing across-process, multiple-project metrics designed to permit monitoring of CMMI compliance. The open source tool Spago4Q has been developed to support our approach and is capable of producing the measurements needed for monitoring of a set of large-scale development projects using different process models, in a real industrial setting in Europe.

1. Introduction

In practice large software companies may undertake projects for disparate application domains which are of varying sizes, delivery schedule requirements, and for different delivery platforms. There may also be joint software development projects with teams consisting of staff from more than one organization (e.g. supplier-client, multiple partners, supplier-outsourcer etc.). This latter form of co-development is becoming increasingly important for large independent software developers.

As a result, many large development companies require competence in several process models to support their activities, even though it may appear that there would be benefits from having a single, company-wide standard software process.

Despite this, establishing a common set of reliable company-wide project performance measures that are independent of the process used is clearly desirable. The objective would be the measurement and comparison, and hence the management of the entire project portfolio. This would be especially desirable, where maintenance of some certified process standard is necessary, and to guarantee uniform levels of process and product quality across all projects.

The development of these frameworks requires some care. Kitchenham, et al [1], reported the case of a major company whose project recording practices made subsequent verification of ongoing maintenance of the certification level challenging.

We describe an open source tool, Spago4Q, designed for monitoring multiple large-scale development projects in a real industrial environment. A Meta-model [2] based upon OMG [3] is used by managers to set up a “measurement campaign”, and to interpret its results.

Existing project management and other tools are used on a flexible COTS basis to both provide data and to perform analysis.

2. Related work on process monitoring

A fundamental problem facing software process engineering is verifying and ensuring that the mandated process is actually carried out. This requires monitoring the actual process steps performed and the work products being produced. The information needed can be obtained either by monitoring actual projects as they are carried out, using some combination of manual and automatic recording, or by careful post-mortem analysis of process enactment.
records. The problem of attaching semantics to the recorded “steps” in a project, is non-trivial. However, post-mortem techniques don’t help a project manager who needs to monitor actual activities as they occur. Online recording and analysis of process enactment data is the only option [4].

An alternative approach involves monitoring software projects by using data on defect-rates, artifact metrics and resource utilization. Early efforts included Victor Basili’s TAME (Tools for a Measurement Environment) [5], and Walt Scacchi’s SoftMan [6].

In [7], Luigi Lavazza described a tool automating the most expensive phases of Goal Question Metrics (GQM) processes. An example of data collection in a specific software product development is presented. To the best of our knowledge, most research efforts focus on single-process, single-project data collection.

There have been a number of research projects and commercial tools which address this problem, although none as far as we aware use a Meta-model, and most are not open source. Holkar [8] (supervised by Torab Torabi) developed a prototype quality monitoring system which used XML as means of both capturing data from sources and storing results in a repository. A limited set of metrics related to budget performance were included in the first version. The Hakystat project at the University of Hawaii [4] provides a measurement framework for non-intrusive project metric recording and analysis, however, this is not reported as meta-model driven. Commercial tools such as Polarion [9] and 6th Sence [10] do not support cross-project comparison and are not metamodel driven.

While a fundamental objective is for the collection activities to be “unobtrusive” most projects, including our own, find that non-automatic recording is needed. This is performed by (other) tools which are then “interrogated” or report to the process monitoring tool.

3. A modular meta-model supporting process monitoring

Our aim is threefold: (i) supporting across-processes data collection, (ii) supporting multiple metrics frameworks, including Goal-Question-Metric (GQM), Balanced Scorecard (BSc), ISO 9000 and Capability Maturity Model Integration (CMMI), (iii) analyzing process and product data at different levels of granularity, supporting company-wide as well as per-process views.

These requirements are met by a meta-model representation of the software process and of the measurement activity [2] having threefold modular structure representing (i) the process(es) to be monitored (ii) the measurement to be taken and (iii) the assessment framework to be used. These three

Fig. 1: Our modular meta-model. The dashed boxes distinguish the Process, Measurement, Trigger and Assessment meta-models.
components have been designed independently, allowing them to be applied the same measurement framework to different processes, or to measure the same process according to different measurement frameworks.

The interface between the process and the measurement modules is the specification of measurable attributes of process activities and work products, needed to compute the metrics. Computing process Key Performance Indicators (KPIs) is done via specific extractors that know which attributes to extract and where these attributes could be found in process and product definitions. Finally, all process metrics to be included in process reports are defined in terms of KPIs.

The meta-model’s components are described using the Object Management Group’s Meta-Object Facility (MOF) notation. Such a standard description supports the interchange and the fusion of data from different sources. Fig. 1 presents our entire meta-model, where the dashed boxes distinguish the individual meta-models, which are briefly reviewed in the following of this section.

Development Process Meta-Model

The development process meta-model has been designed to be as generic as possible, allowing for modeling virtually all process models from waterfall to XP. It is loosely based on OMG’s SPEM (Software Process Engineering Meta-model) specification. We selected only those SPEM entities strictly needed for process monitoring [11]. The top element is the node Organization, which gathers all the Project nodes. Each project is associated to a specific development process, defined as a set of Phase meta-classes, which can be linked to other phases to allow iterative development paradigms, and composed by a set of Activity nodes. Depending on the level of granularity chosen, activities could be decomposed in Task and Action nodes; here, for the sake of conciseness, we limit our analysis to activity class.

The results of the work performed in each activity is described by Work Product elements, while the nodes Role and Tool define the roles that carry out a specific work product and the tools used to produce it.

At the same time, the Project node is connected to the Product meta-class, that describes the type of product under development and to the Customer node, that identifies the customers’ type and allows its classification w.r.t. their business activity.

Measurement Framework meta-model

The Measurement Framework meta-model defines a skeletal generic framework exploitable to get measures from most development processes.

The Information Need node is a container that defines the need that drives all measurement actions. For each information need there is a set of Measurable Concept nodes defining the areas of analysis. Examples of Measurable Concept instances include “Requirement management” or “Software quality”. These concepts drive the definition of Measurable Attributes to be measured in order to accomplish the analysis goals. Furthermore, this element specifies how attribute values must be collected. The Measure node defines the structure of data retrieved during a measurement campaign, supplying raw data to KPIs. Each node is specified by Scale Type and Unit classes, defining, respectively, the unit of measurement used and the type of scale adopted (nominal, ordinal, and so forth). The monitoring indicators are defined in terms of KPI/Metric. KPIs identify simple transformations reducing the cardinality of measures, e.g. eliminating outliers. In turn, KPI values are used as input for Metric, which defines conditioning and pre-processing of measurements in order to provide meaningful indicators. Finally, the Metric class is in relation with the Threshold entity specifying the threshold values.

Assessment Meta-Model

The Assessment meta-model allows a simple classification in terms of Category, Target, and Practice. The Practice node is connected with the Metric class in the Measurement module, highlighting the concept that the evaluation of a set of metrics could assess the real appraisal of the practices. The strict correlation between assessment framework practices and metrics will be clarified in Section 4.

Trigger Meta-Model

The Trigger meta-model formalizes and defines the extractors used to retrieve data from process module and supply them to measurement modules. The module is composed by two meta-classes, Trigger and Triggered Value, respectively describing the extractors themselves and the repository of extracted data.

Triggers are connected with all the elements of development process meta-model that are involved in the measurements regime, and they are connected with measurable attributes, indicating which attributes have to be measured. Triggers operate in strict relation with tool through the exploitation of extractors and specific tool plug-ins. These components allow the interaction with the tool that generated the work-products to be
measured. In some cases, extractors could directly access to data without interacting with plug-ins, like for instance retrieving data about the number or the size of java classes produced.

Triggers achieve independence between the process and the measurement meta-models: triggers know which data to extract, defined by measurable attributes, and they are configured ad-hoc by project managers in terms of extractors and plug-ins, in order to physically retrieve them. Thanks to a rigorous configuration work, it is possible to apply the same measurement framework to multiple process instances.

4. CMMI-based Assessment

While the current implementation of our approach (Section 6) supports a number of development processes, our current assessment module is instantiated as a single set of KPIs and metrics based on CMMI. Since CMMI does not specify which metrics to collect, the method of collection to exploit, and nor the appropriate interpretation of the measured data, we took its goals and practices as the basis to a complete GQM process [4] for the evaluation of a software process. Further, our search revealed many exhortations to “measure” as part of CMMI adoption, but we were unable to much in the way of guidance. Perhaps Palza [12] and Xu [13] provide the best examples and advice. CMMI does, however, contain all the information needed to establish the goals of a measurement campaign and the objectives (questions) which must be resolved in order to achieve the goals [14]. This idea has been already explored, for example, by Järvinen et al. [15], who used the results of process assessment as metrics. In this way, a standard assessment framework can be adopted as quality model, allowing the evaluation of both high-level characteristics, and low level attributes of running processes. In order to produce a GQM for CMMI, we followed the three-step procedure described below.

Firstly, a set of goals were defined using the CMMI assessment model, in particular focusing on the objectives defined in each CMMI process area. These objectives can be the goals defined by the process areas, or some intrinsic characteristics of the process area that the goals aim to describe and control. For instance, looking at the Requirement Management CMMI process area, we can take as a measurement goal the Specific Goal 1 “Manage requirements”, or focus on a different requirements aspect of development process, that is one described within the other goals of the CMMI Requirement Development process area. This is important when the aim of the measurement campaign is to control some process maturity features, examined under a CMMI perspective, but not directly to assess CMMI compliance.

Secondly, we identified a set of measurable attributes that a single question should measure to produce an actual estimate of a CMMI goal. For instance, again taking into account CMMI practice 1.3 “Manage requirements changes”, which address the requirement stability problem, we considered, as a simplified stability attribute, the requirement state (as for instance stable, new, modified, implemented). The corresponding KPIs are the number of requirements in each of the states.

Finally, we defined metrics for each question. It is important to note that, for each metrics, a threshold is to be defined in order to determine if a metric value satisfies or not the underlying CMMI practice. In the requirement stability case, a simple metric could be “Requirement stability” computed as the number of changes, which is the sum of modified plus deleted plus new divided by the total number of requirements1. In [16] a complete list of CMMI-based KPIs and metrics are provided.

5. Computing across-processes metrics

As we pointed out in our introduction, the combination of co-development, multiple domain/process/platform attributes of the projects undertaken by a large software company make multiple software process competence and practice essential. Despite this, they need to compute company-wide (as opposed to process-specific) process and product quality indicators. Our meta-model approach is suitable for cross-process measurements, but it is important to establish which metrics and KPIs (coming from different projects) can be safely aggregated and which ones cannot due to semantic differences. Indeed, it would not make sense to compute metrics by merging arbitrary KPIs coming form measurements on, say, RUP requirements with ones measured on XP user stories, even if both work products are represented by the same entity in the process module of our meta-model. Our preliminary experimental results confirm this (see section 7).

Basically, there are two ways to handle this problem. The first one is applicable when a developer adopts variations of the same development process (e.g., different Crystal colors or Scrum vs. Waterfall); in this case, all process variations share work products whose attributes have similar semantics; the corresponding KPIs and metrics can be safely merged

---

1 A more extensive set of statistical measures can of course be used.
to compute cross-process metrics. More frequently, however, developers have to adopt customer-specific processes, e.g., using RUP for a customer and XP for another one. In this case, explicit semantics annotations are needed to identify which KPIs and metrics can be merged to compute across-processes metrics.

As shown in Section 7, explicit support for relationships between KPIs and metrics of different processes is a basic requirement for a platform supporting our approach. Also, in this case aggregation must be handled with care, avoiding average-based aggregations (which could compensate poor performance in a project with better performance in another one) in favor of computing metrics over KPI ranges, obtaining min/max metrics intervals. Section 7 contains some min/max aggregation examples.

Despite this, there may be quality (i.e., final product defect rates), and productivity related metrics that can be meaningfully merged.

An interesting and important requirement of CMMI is to be able to show improvement. Our view is that the design of a metrics program which deals with this is non-trivial. Consider for example, the initial negative productivity and quality impacts of adopting a new tool or sub-process methodology. Obviously, these should not be deployed until equivalent levels of competence have been achieved, so negative outcomes from an “improvement” activity could indicate bad adoption practice. In our case, temporal aspects can be introduced to measure changes in individual process competence and its impact on performance KPI’s.

6. The Spago4Q Platform

Spago4Q (SpagoBI for Quality) [17] is a Free Open Source platform that fully adopts the proposed meta-model approach. It can be easily adapted to complex organizational contexts, independently from the adopted software development processes, infrastructure tools, measurement and assessment frameworks. Spago4Q is suitable for maturity assessment, effectiveness of development software process and quality inspection of the released software: this goal is achieved by evaluating data and measures collected from the project management and development tools with non-invasive techniques. From an architectural point of view (Fig. 2), the ETL (Extract Transform Load) procedures extract data from the set of tools used by the organization and load them into the DWH-Spago4Q data warehouse module, based on the meta-model definition; Spago4Q analyzes data and represents metrics, and the Configuration & Administration module allows system configuration.

A set of predefined plug-ins and extractors allows the extraction of data from the most common project management repository (e.g., MSOffice and OpenOffice documents), artifacts repository (SVN, CVS, etc.), and development tools (like for instance Eclipse).

Spago4Q supports CMMI as its standard assessment framework. It provides advanced reporting capabilities, including defining metrics on semantically

![Fig. 2: Spago4Q architectural overview](image-url)
equivalent KPIs. The keeping of CMMI practices is monitored by specific KPIs and metrics, defined following the roadmap described in Section 4, examining their values with respect to predefined thresholds, Spago4Q gives at a glance a reliable snapshot of across-process indicators and allows managers and assessors to monitor the adherence of CMMI goals [18].

Spago4Q has the following main characteristics. Firstly, it is an open source product, that could be freely adopted from any organizations. Secondly, the metamodel approach allows real cross-process measurement regimes, since the same measurement regimes can be applied to projects following different developing paradigms and, under some circumstances, data from these measurements could be gathered and analyzed together. Finally, since the application is released as open source, it is possible to build organization-specific extractors to deal with particular file formats.

As we have already said, Spago4Q’s ability to monitor projects in organizations following different development approaches is due to the adoption of the meta-model approach. It drives the definition of data warehouse, the relations between tables, and the relations between the objects of the framework.

Looking at the tool architecture, there is a slightly difference between the definition of the specific development processes and measurement frameworks, and of the data warehouse that stores the measured data.

In the first case, any node of the metamodels is considered as a table of two different databases (Development Process and Measurement). Tables contain fields that describe any element of the metamodel, and provide information that could be useful in analysis actions.

In the second case, data are extracted and stored directly in the data warehouse. In that, nodes of the process metamodel become dimensions of the data warehouse. This allows exploiting OLAP analyses under any aspects of the meta-model (process, phase, activity, etc.).

At specific time intervals, Spago4Q calculates KPIs, retrieving all the information needed for the calculus from KPI table, and extracts data from the data warehouse, analyzing it using as analysis dimensions data specified in that tables. As for instance, if we need the requirement number for the project X, we traverse the data warehouse extracting data relative to the dimension “Project” equal to “X” and “Work Product” equal to “requirement”, and then applying to the returned dataset all the needed computation, specified in the KPI table as an external procedure with the specific calculation task.

As stated above, it is possible for organization to build specific extractors. The implementation of them could be easily performed by the integration of a provided Spago4Q library and the implementation of a specific interface.

7. Experimental results

The experimentation was carried out at Engineering Ingegneria Informatica [19].

The different markets where the company operates, the heterogeneous technological environments, and diverse contractual agreements, mean that the company must have competence in multiple development environments and software practices and must adapt them to specific projects. In a continuous Software Process Improvement context, the company has to manage and control multiple single software projects to achieve the projects’ goals and, at the same time, improve the company’s software processes as a whole. Every project uses a specific software development process adapted to the customer’s requirements. At the same time, the company uses the meta-model approach outlined in Section 3 in order to develop a single environment for “measure collection”, performance analysis and processes adjustment.
The experiments was carried out on seven pilot projects. Data from three are reported here. The development processes were Waterfall, EWebMo (an ad-hoc evolutionary-iterative software process model by Engineering), and Scrum. In this section, we focus on monitoring metrics related to the requirement management goal. We collected data about requirement definition and tracking, testing, and issue/bug tracking phases; in the following we report results about the requirements variability. Fig. 3 shows requirement variability trends of all three projects for a six-month period.

Project 1 was a new development for a large life-insurance company; it was organized in sub-projects (with a 6-months predefined duration) using EWebMo as development process. We report on requirement phase of one sub-project which lasted 4 months. The first month is spent for the definition of requirements baseline document. A monthly customer validation and internal peer-review of the document is conducted. Project 2 was a new system for a logistic company; it lasted 18 months and used a waterfall development process. The requirements management phase is similar to that of Project 1, but had bi-monthly reviews. Project 3 developed an information system for handling multimedia data in a healthcare company. The development process was Scrum with a Sprint review each month.

Project 1 requirements variation is under the predefined 10% threshold set as the company’s target. It is a sub-project of a “known” development process used in a multi-year incremental delivery project with periodic peer-reviews (only few modified requirements). The variation was expected to be near to 5%, but the cancellation of two requirements in the last three months has raised this to about 8%. Also Project 2 has a low rate of requirements variation, due to the level of detail of the main technical project description and to the well-known development environment.

Project 3, which follows an agile approach, is characterized by a far higher variability. In fact, only the requirements variations of the Sprints following the first backlog have been collected. Fig. 4 shows cross-project aggregation of requirements variability using three simple aggregators: Min, the minimum value of each curve, Max, the maximum, and Mean, the average value are calculated from the three graphs. In general, the requirements variation rate is under the predefined company threshold in all projects except for the Agile project, while the mean (as it is a compensative aggregator) could lead us to incorrect conclusions.

The company-wide view of across-process variability of requirements plays an important role, since the underlying contractual agreements mandates high-level of requirements compliance in a predefined time-scale. The Max aggregation shows how this company-wide factor compares process-specific ones, suggesting that risks related to contractual obligations on requirements is under control company-wide.

8. Conclusions and Further Work

When multiple processes are involved, traditional process analysis practices mostly relied on qualitative
judgment in order to avoid tricky aggregation of semantically heterogeneous metrics. Today, co-development with customers and outsourcing are powerful drivers toward adopting multiple processes, and require an effective scientific methodology for company-wide, across-processes assessment. We have shown how a simple meta-model approach can integrate quantitative analyses of different processes, increasing the effectiveness of company-wide risk detection and mitigation practice.

Our technique can help in identifying the areas for improvements, and develop a mechanism for assessing company-level distance from maturity goals. We also aim to identify the processes which are causing most of the problems and suggest strategy to eliminate or reduce the harmful effect of these processes in minimum cost and time.

Acknowledgements

This work was partly founded by European commission under the project SecureSCM (contract n. FP7-213531) and by the Italian Ministry of Research under FIRB TEKNE (contract n. RBNE05FKZ2_004).

References


