Development of a Unified Software Quality Platform in the Szeged InfoPólus Cluster

Árpád Beszédes
Department of Software Engineering
University of Szeged
Szeged, Hungary
beszedes@inf.u-szeged.hu

Lajos Schrettner
InfoPólus 2009 Ltd.
Szeged, Hungary
schrettner@infopolus.hu

Tibor Gyimóthy
Department of Software Engineering
University of Szeged
Szeged, Hungary
gyimothy@inf.u-szeged.hu

***Abstract*—In Software Quality Assurance, system status reports are key to the stakeholders for controlling software quality. Although we may rely on a large number of low level measurements to this end, their interpretation and the way of connecting them to high level quality attributes is always a challenge. In this paper we report on a complex project involving industrial partners whose aim is the development of a unified software quality platform that deals with and bridges these low and high level quality aspects, and provides a basis for the industrial applications of the approach. The project is implemented by a consortium of software industry members of the Szeged Software Innovation Pole Cluster and associated researchers with the support from the EU co-financed national grant promoting innovation clusters of small and middle-sized enterprises. The approach to the unified quality platform is based on the Goal-Question-Metric paradigm and a supporting software infrastructure, and its novelty lies in a unified representation of the low level metrics and the high level questions that evaluate them to address software quality assurance goals. It allows the definition of various complex questions involving different types of metrics, while the supporting framework enables the automatic collection of the metrics and the calculation of the answers to the questions. We present details about the design and development of the quality platform and its relation to the applications that are being developed by the industrial members of the consortium.**

***Index Terms*—Software quality assurance, Metrics, Goal Question Metric, Quality model, Modeling.**

I. Introduction

The Software Innovation Pole Cluster [1] was formed as the result of collaboration of major software research centres, companies and technology transfer organizations. Based upon earlier cooperations the members of the cluster realized that they can produce more competitive technologies, products and services if their developments are harmonized. By extending collaborations and by more efficient coordination of cooperations the cluster offers its members the consolidation of high added value R&D activities with related training and education, as well as the creation of the infrastructure for knowledge intensive activities.

The SIKK (Hungarian abbreviation for *Innovative Research Center for Software Industry*) initiative is organised around the Software Engineering Department at the University of Szeged and brings together a number of local companies that are willing and able to benefit from sharing their experiences and expertise in the field. The center has been growing dynamically over the past years, and has produced a number of results that are readily available to the partners for use in their products and services.

Software quality is of prime importance to every software company, and it lies at the center of interest for the Software Innovation Pole Cluster and SIKK members as well, so they formed a legal entity, InfoPólus 2009 Ltd., to create a formal context for cooperation in a project. The motivation behind the project is the realization that without constant quality monitoring the number of errors in a software system tends to rise over time. It is inevitable that systems are modified from time to time due to scheduled releases, bugfixes, changes in the requirements and other reasons. Unwanted side-effects of modifications usually cause a sudden increase in the number of errors, most of which are gradually eliminated, but studies show that the total number of errors in a system exhibits a steady increase [2].

Quality monitoring is traditionally based on measuring low level characteristics (metrics) of the system, but to be really useful it requires a number of well-constructed quality goals to be formulated as well. It is still a challenge, however, to close the conceptual gap, i.e. to establish a relationship between the low level metrics and the higher level goals. A bottom-up approach has several drawbacks because there are many characteristics in software (e.g. lines of code, complexity, number of bugs), but selecting the important ones is not straightforward without well defined top-level goals. The Goal-Question-Metric (GQM) paradigm [3][4] proposes to narrow the gap by introducing an intermediate level, i.e. questions that help in achieving quality goals in a top-down fashion by combining a well defined set of metric values.

Our proposed solution to quality monitoring is based on GQM in the sense that we suggest a method for combining metrics and questions in a unified model (see Figure 1). The process starts with analysis steps formulating quality goals, which are translated into questions that in turn can refer to metrics. The result of this analysis is a data model that describes the set of metrics to be measured and the set of (parameterized) questions that can be asked to achieve the goals. We found that the model is best organized by partitioning it in a way that reflects the lifecycle phases observed. Also, it may

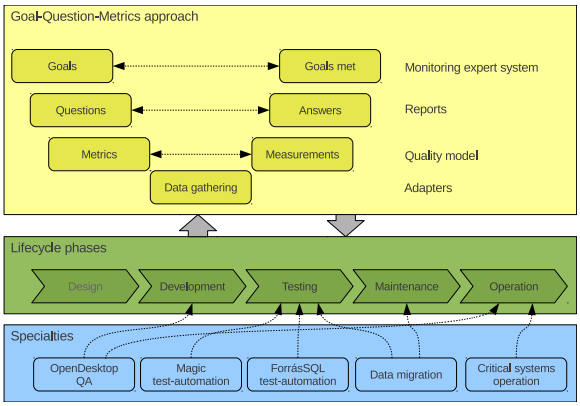

Figure 1. Unified Quality Model Platform design principles

be advantageous to include extra metrics on top of the ones that are strictly required to answer the questions, because this makes it easier to extend the set of questions at a later time. The tools of each specialty feed data into the model instance through adapters. Quality goals can be met by activating the questions once enough metric data are accumulated.

In this paper we first overview the project in Section II, then summarize the project goals in Section III. The current status of the project is described in Section IV, while plans for the future appear in Section V.

II. The Project

The project is carried out in cooperation between the parties below:

- **Büro-Three Ltd.** was founded in 2008, it has been working as a sub-contractor on the development of the information system of the Hungarian Trade Bank (MKB). The tasks of the company are the completion of certain error corrections after the installation of MKB-Globus system, fulfillment of commercial warranty and support of help-desk service.
- **Clarity Consulting Ltd.** [5] has an extensive IT and management consulting experience gained by working with one of Hungary's largest IT consulting companies. The company is built on the belief that in order to quickly find the most effective business solutions for its clients, one must have practical experience and a solid grounding in business models and IT solutions for their particular line of business.
- **FrontEndART Ltd.** [6] explores, develops and markets innovative solutions for software quality management. The most popular product and service of the company is the software quality assurance system based on Columbus technology, instrument and quality measurement service.
- **Griffsoft Informatics Plc.** [7] is a business management software developer and IT services provider company in Hungary. It has been developing and selling the Forrás management system for more than 20 years. Development activities center around ForrásSQL, a high level database-oriented programming language of the company.
- **MultiRáció Ltd.** [8] was founded in 1992 for the design, development and operation of economical and financial analysis systems. It won an innovation prize in 2003 for the development of the OpenOffice.org commercial derivative called MagyarOffice and later EuroOffice. Besides analytical systems, MultiRáció also offers solutions for other problems that require special expertise in areas of statistics, mathematics, informatics or finances.
- **Polygon Informatics Ltd.** [9] is the largest Hungarian Premier Business Partner of IBM, its main profile is the selling and popularizing IBM technologies. The company also deals with server operation, the integration of systems, servers, networks and applications, and it has significant sales revenue from selling hardware as well.
- **Szeged Software Plc.** [10] has been developing applications which satisfy special information claims of wholesalers since its foundation. It developed PharmaLog for the pharmaceutical wholesalers, and StoreIS, a general, wholesale logistic system.

Details of the project are the following:

- The project is based on the Support for market-oriented research and development tender in the Economic Development Operational Programme, New Hungary Development Plan. The project is supported by the European Union and by the European Regional Development Fund.
- The full title of the project is *Development of a unified software quality platform and business applications based on the competence of the SIKK knowledge center and the development and market expertise of the participating members of the Szeged InfoPólus*, its registration index is GOP-1.2.1-08-2009-0005.
- The total budget of the project is 5 620 000 EUR. The amount of funding is 2 810 000 EUR (50%).
- The project is carried out between 1 October, 2009 and 30 September, 2012.

The Software Engineering Department at the University of Szeged plays a significant role in several aspects, it supplies its expertise for designing the conceptual background for the project and the supporting infrastucture for the quality monitoring platform. The most important result of the SIKK initiative is the RÉM (Hungarian abbreviation for *Software Development Life Cycle Methodology*), which is a constantly maturing, unique and comprehensive software quality assurance model. It is applicable not only for determining software quality, but for monitoring and controlling it in a unified framework. The main element of the concept is the need for continuous centralized tracking of quality attributes, i.e. measurements. A particular application of the concept requires the determination of a tailored set of quality attributes, so that continuous control over them could put a halt to system degradation.

III. Goals of the Project

Reflecting the fields of expertise of the individual members of InfoPólus 2009, the applications developed in the project

and whose quality is to be monitored are quite diverse. The main goal is to design and develop a common software quality measurement platform to which the individual applications can connect despite the diversity.

The applications developed by the partners have been assigned to so called *specialties* to place them in a wider context. The use of specialties emphasizes that the results obtained can be generalised to application areas similar to the ones found in the project.

The following specialties have been established:

- **OpenDesktop open source quality assurance**. The goal is to create a high quality, fully functional office platform and services built from open source components using IBM technology. Further, the platform is to be supplemented with software quality assurance methodologies, tools and services that help in selecting desktop applications to be included in the distribution.
  Key members: MultiRáció, FrontEndART.
- **Magic test-automation**. The goal is to develop a complex product and services bundle to facilitate the more effective and reliable testing of Magic systems by using test automation.
  Key member: Szeged Software.
- **FrorásSQL test-automation**. The goal is to develop FrorásSQL-specific tools to automate the generation and execution of tests using databases that contain millions of records, thereby simulating real-life environments.
  Key members: GriffSoft, FrontEndART.
- **Data cleaning**. The goal is to develop a set of data manipulation tools that make the maintenance and correction of real and test data possible. One important area is data masking where test data can be generated from real data by a method that alters sensitive values but keeps consistency and associations.
  Key member: Clarity.
- **Safety critical operation of large systems**. The goal is to develop an application and services for the monitoring of large, safety-critical systems that exhibits functionality beyond the capabilities of other monitoring tools. This is achieved by handling behavior that is specific to the observed system instead of registering general hardware-, database-, and environment-related events only.
  Key members: Büro-Three, Polygon.

The applications and tools above cover a wide spectrum of the software development life cycle. The members can expect a number of benefits from the project:

- The availability of a unified quality platform which they can include in their products.
- Specialized quality assurance solutions by customizing the platform.
- Added value to products, services offered.
- Potential of increased market share.
- Continuous measurement and evaluation.
- More efficient IT operation.

IV. Status of the Project

The project has reached a phase when the concepts, architectural design documents and prototypes are available. The tools belonging to the specialties cannot be dealt with here due to space constraints, so we concentrate on the framework itself in the rest of the paper. At present the details of the user interface are under development, in which each partner is involved.

The Unified Software Quality Platform that integrates the quality data collection, high level query management and reporting features has been implemented on top of our MBPS framework. The components have been designed and implemented to be as general as possible so that they can be reused in other projects as well.

A. Model Based Persistence Server

The Model Based Persistence Server (MBPS) is capable of supporting clients that benefit from viewing their persistent data as a collection of nodes (and connections) instead of as records of a relational database. Its main feature is that besides serving as a vehicle for model-based data storage, it supports the integration of queries into the models.

The structure of the data to be stored is defined via a UML model description that is processed to create model instances. The framework provides a model-independent interface that can be used to retrieve, insert, delete, and modify model elements, as well as executing queries. The UML description of a model can also serve as a basis to generate a model-specific API that can be used either on the client side or the server side. The framework uses an internal cache to speed up both data store and retrieval operations.

During building and manipulating the model instance, the framework continuously checks that the instance is kept consistent with the model, it only allows operations that do not invalidate consistency. Model instance elements are versioned, so modification histories are available, even queries can refer to past versions of elements, which is very convenient as historical diagrams are needed very often in quality monitoring.

It is possible to upload application-specific Java code to any domain, so it can run on the server side, thereby improving efficiency and supporting post-processing of query results.

Uploaded custom Java code has two main uses:

- It can improve efficiency by processing data at the server side, because less data have to be communicated between the client and the server. This feature comes very handy in cases when efficiency is a top priority.
- It can be used to add post-processing steps to queries. The query model does not allow for any possible type of aggregation of data to be performed, so sometimes it is necessary to run custom code on the query results.

B. Unified Software Quality Platform

The platform is still under development, but it already provides limited functionality. Quality data accumulates in the model instance by the use of special client components, called adapters (Figure 2). Adapters are data transformation devices

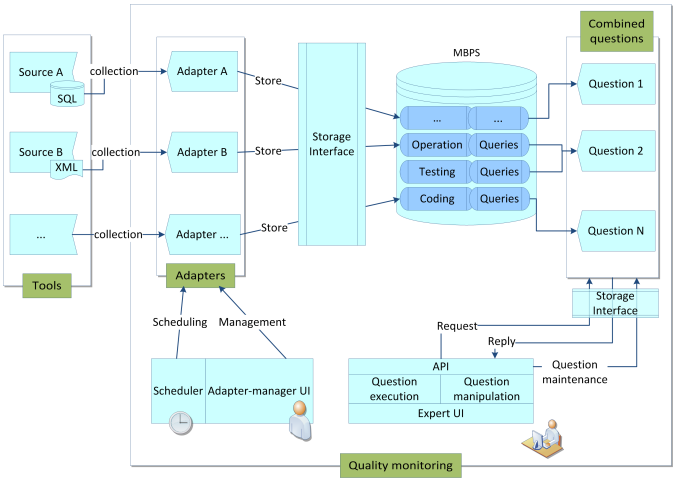

Figure 2. USQP architecture

that collect data from various sources, restructure them as dictated by the model, then forward the resulting elements to the persistence server. There are a predefined set of adapter types that are prepared to be able to collect data from sources that have been encountered so far in different projects. Adapter types and supporting classes are arranged in an inheritance hierarchy, so new adapter types can be included into the system relatively easily if the need arises. In a particular installation of the Unified Software Quality Platform application, those adapters that are determined to be necessary for the operation of the system have to be instantiated. Most adapters require that parameters be provided at instantiation time to connect to their data source. Instantiated adapters can be set up to operate under the supervision of a scheduler that activates them at preconfigured moments, or they can be activated manually.

Data collected through adapters accumulates in the persistent storage, where it can be queried from. Queries are stored next to the model in the supporting database, and can have parameters that should be filled in before execution. The parameters together with the constraints that are described inside the queries determine a subset of the nodes and connections of the model instance. This way they are similar to relational queries, except that they produce a subgraph, not a series of records. Queries can encode a limited number of aggregation functions (e.g. count, sum, average), but these do not cover every possible usage scenario, so it is possible to attach custom post-processing Java code to queries, as mentioned above. The results of a query are sent to the client, where it can be decided how it is stored or presented to the user. As mentioned earlier, queries are stored next to the model instance, where they can be efficiently executed. The results of an executed query can be viewed textually in the simplest case, or they can appear on a diagram if the output type of the query is one of those for which diagrams are predefined. The current set of diagrams consists of several two and three-dimensional bar charts, timelines and other reports. There are three views in the unified user interface of the system, two for administrative tasks (Adapter management, Question manipulation) and another one for the end users, i.e. the experts who would like to monitor the quality of the observed system(s).

There are typical usage scenarios for the Unified Software Quality Platform:

- **Single source.** In this case there is one observed system on which one measurement is carried out only, questions can only refer to the metric data of that same measurement. This is the simplest and most straightforward use of the platform.
- **Multiple independent sources.** In this case there are several observed systems that are independent from each other in the sense that each sends data to a separate measurement part in the model. Questions refer to the metric data that belong to a single measurement, i.e. to a single observed system.
- **Multiple sources.** This scenario is the most interesting from a software quality point of view, because here we use multiple measurements along with questions that require data from more than one measurement. The data sources in this case are subsystems that are operated together to serve a common goal.

V. Future Work

At present the platform is set up in an evaluation environment where partners can conduct tests on real data collected from the applications. The user interface of the platform and the underlying models are going to be developed further based on feedback from the evaluation period. After incorporating the requested features, the USQP platform and tool can be included in the tool packages of the individual specialties.

Acknowledgement

This research was supported by the Hungarian national grant GOP-1.2.1-08-2009-0005.

References

[1] "The InfoPólus cluster homepage." [Online]. Available: http://www.infopolus.hu/
[2] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. McGraw-Hill Higher Education, 2001.
[3] V. R. Basili, G. Caldiera, and H. D. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*. Wiley, 1994.
[4] V. R. Basili, "Software modeling and measurement: the Goal/Question/Metric paradigm," College Park, MD, USA, Tech. Rep., 1992.
[5] "Clarity Consulting homepage." [Online]. Available: http://www.clarity.hu/site/en/index.php
[6] "FrontEndART homepage." [Online]. Available: http://www.frontendart.com
[7] "GriffSoft homepage." [Online]. Available: http://www.griffsoft.hu
[8] "MultiRáció homepage." [Online]. Available: http://www.multiracio.hu
[9] "Polygon homepage." [Online]. Available: http://www.polygon.hu
[10] "Szeged Software homepage." [Online]. Available: http://www.szegedsw.hu