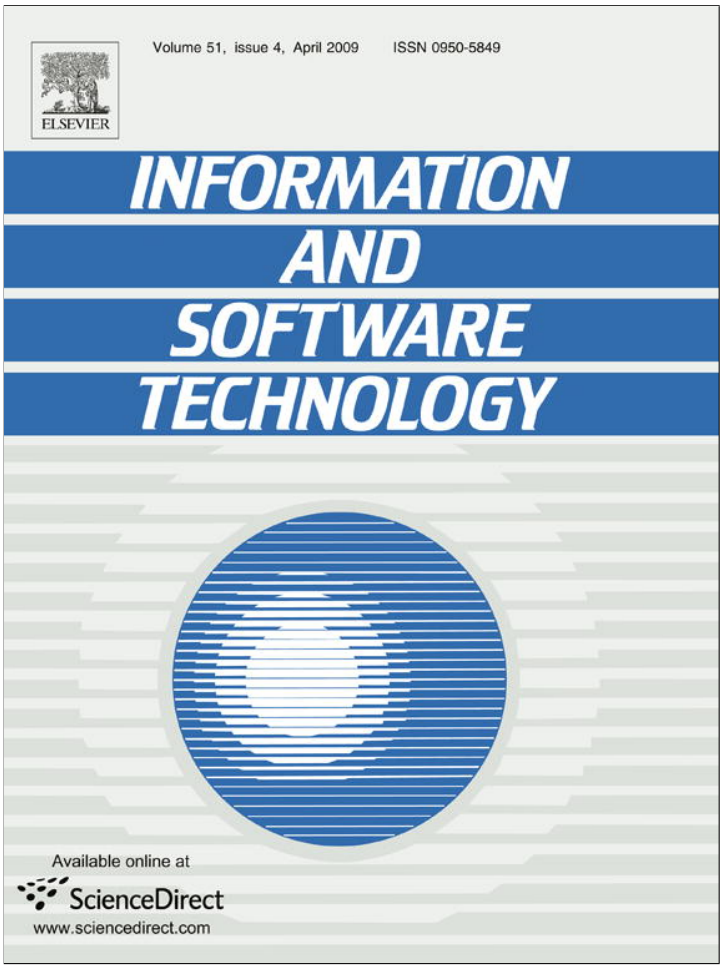

Information and Software Technology 51 (2009) 721–737

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A framework for developing measurement systems and its industrial evaluation

Miroslaw Staron [a,*], Wilhelm Meding [b], Christer Nilsson [b]

[a] IT University of Göteborg, SE-412 96 Göteborg, Sweden
[b] Ericsson SW Research, Ericsson AB, Sweden

ARTICLE INFO

Article history:
Received 25 February 2008
Received in revised form 15 September 2008
Accepted 28 October 2008
Available online 5 December 2008

Keywords:
Software metrics
ISO/IEC 15939
Measurement systems

ABSTRACT

As in every engineering discipline, metrics play an important role in software development, with the difference that almost all software projects need the customization of metrics used. In other engineering disciplines, the notion of a measurement system (i.e. a tool used to collect, calculate, and report quantitative data) is well known and defined, whereas it is not as widely used in software engineering. In this paper we present a framework for developing custom measurement systems and its industrial evaluation in a software development unit within Ericsson. The results include the framework for designing measurement systems and its evaluation in real life projects at the company. The results show that with the help of ISO/IEC standards, measurement systems can be effectively used in software industry and that the presented framework improves the way of working with metrics. This paper contributes with the presentation of how automation of metrics collection and processing can be successfully introduced into a large organization and shows the benefits of it: increased efficiency of metrics collection, increased adoption of metrics in the organization, independence from individuals and standardized nomenclature for metrics in the organization.

1. Introduction

In a large software development organization metrics collection and analysis is an important part of daily work. In such organizations, large numbers of software metrics are collected and analyzed in order to monitor organizations, projects, and products, leading in the long run to optimizations and increased operational excellence. Metrics collection and analysis, however, requires significant effort to effectively present the results of measurement processes. A way of optimizing measurement processes is to use custom measurement systems, which collect information from several sources, automatically analyze it and inform the user (stakeholder) about the key status indicators or problems. Despite numerous advantages, developing such custom measurement systems requires a significant amount of time as standard metrics need to be adapted to the concrete information needs which the measurement systems should fulfill [1,2].

By using measurement systems the managers, can focus on a few key indicators (metrics with associated interpretations) instead of monitoring large number of metrics. This, in consequence leads to increased performance of managers and efficiency of their projects and organizations. Furthermore, using comparable measurement systems in multiple projects, can lead to easier benchmarking of projects or even organizations; naturally if the metrics have appropriate definitions (as indicated in [3–5]).

This paper presents a framework which addresses the need of quick development and creation of measurement systems. The framework implements ISO/IEC 15939 standard, in particular its information model, and after instantiation forms a measurement system which satisfies information needs of its stakeholder (e.g. product manager's need to monitor post-release product reliability) by collecting, analyzing and presenting data from measurement instruments. These measurement instruments are tools used to obtain a value of a metric; a task which in software engineering is performed by "metric tools". As effective use of these measurement instruments in decision support processes requires combining the resulting values and presenting them as key indicators, measurement systems combine metric values, interpret them using pre-defined decision criteria and later present the indicators to the stakeholders.

The main purpose of the framework is to make the work with metrics more efficient and effective in large organizations. The framework presented in this paper is primarily dedicated for engineers and managers working with metrics collection and data analysis. These tasks are supported by the mechanisms built into the framework: pre-defined kinds of metrics (base metrics, derived metrics, and indicators as defined in Appendix A), methods for calculating derived metrics and indicators, and methods for assigning interpretation to indicators. The framework is a pre-prepared MS Excel file with Visual Basic for Applications (VBA) scripts automat-

* Corresponding author. Tel.: +46 31 772 1081.
E-mail address: Miroslaw.Staron@ituniv.se (M. Staron).

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.10.001

ing the process of data collection, analysis and customizing the resulting measurement system. The framework is instantiated by using built-in wizards to add metrics of the kinds defined in ISO/IEC 15939, links between these metrics and decision criteria for interpreting the indicators. The instantiated framework is an MS Excel file which automatically calculates and interprets indicators from the data provided by metric tools or from databases.

The framework is evaluated in this paper through its use in one of the development units with several hundred employees within Ericsson. Using the framework does not require programming (for most tasks) and therefore makes the development and use of measurement systems much simpler task than collecting and analyzing the same data manually.

In addition to presenting the framework, this paper also contributes with the presentation of how automation of metrics collection and processing can be successfully introduced into a large organization through the use of a framework based on ISO/IEC 15939 standard (exemplified by our framework). The paper shows benefits of such a framework: increased efficiency of metrics collection (no manual work is required for daily collection of metrics), increased adoption of metrics in the organization (daily updates of information, minimized effort required to set-up measurement systems, customization for stakeholders), independence from metric experts and standardized nomenclature for metrics in the organization based on the ISO/IEC standards.

The remaining of this paper is structured as follows. Section 2 discusses the most related work to this research, which influenced the existing shape of the framework and its introduction into the company. Section 3 describes the organizational context of our research – the development unit at Ericsson, its history of working with metrics, and their current metric program. Section 4 outlines the design of our study which resulted in the development of the framework. Section 5 presents the framework itself, while Section 6 presents an example of how this framework is used. Section 7 shows the evaluation of the framework and the benefits observed when using it at Ericsson. Section 8 presents threats to validity of the study and Section 9 summarizes the contributions of our research. Finally Section 10 presents conclusions.

2. Related work

2.1. Previous empirical studies of metric programs

Our framework for constructing measurement systems is intended to support organizations establishing metric programs. We investigated the following publications in order to elicit factors important when introducing metric programs into organizations in general, and not to be constrained only to Ericsson's context:

- Umarji and Emurian [6]: the study describes the use of technology adoption theory when implementing metric programs with focus on social issues. One of the important results from that study was the importance of the factor "ease of use". When developing our framework we invested in making the framework easy to use and making the presentation of the indicators easy to interpret.
- Gopal et al. [7,8]: these studies present results and conclusions from a survey about metric program implementation conducted with managers at various levels (over 200 data points). The results indicated the importance of such factors as management commitment and the relative low importance of such factors as data collection. In order to check how important the framework is for the managers who we work with, we included the line manager and the project manager in our interviews when evaluating the framework.
- Atkins et al. [9]: among other aspects, this paper discusses how metrics can be reused by projects working on similar things in parallel. We used their experiences when reasoning about the reuse of metrics between different instances of the framework.
- Lawler and Kitchenham [10]: based on the experiences of several case studies, this paper discusses the issues of using metrics at different levels and combining metrics together (e.g. combining metrics from particular designers to provide the status of the whole project). This work affected the design of the framework in such a way that the metrics in the framework can be reused and combined in a way consistent with the study by Lawler and Kitchenham.
- Kilpi [11]: this paper describes how a metric program was implemented at Nokia. We used their experiences when evaluating the framework.
- Niessink and van Vliet [12 and 13]: these studies describe external factors important for software metric implementation, including the importance of the goal of software measurement processes. Our experiences support this conclusion, and the need for the monitoring status and progress resulted in finally choosing the ISO/IEC 15939 standard as a basis for our work with metrics.
- de Panfilisi et al. [14]: this study describes experiences from introducing a GQM-based metric program. Our experiences showed slightly contradicting picture that one of the most important aspects is not the sole moment of adoption of a program (as advocated by GQM) and possibilities of using subjective metrics, but the use of objective metrics to monitor entities over longer periods of time. A more detailed guidelines supporting the introduction of metric programs can be found in Goodman [15] or Möller [16].

2.2. Other approaches

We deliberately chose the ISO/IEC 15939 (:2002 in the start of the study and :2007 when the study concluded) in our work as the main standard. It is not the only standard (others are mentioned in the appendix), but it is an accepted one and was best-suited for our purposes, which were: (i) need for constant monitoring of status and progress; (ii) need for specification of how to set up infrastructure of measurement systems; (iii) ability to define main attributes; and (iv) a requirement from Ericsson to use an international standard. A good comparison of other related measurement frameworks and models was conducted by Chirinos et al. [17], who considered such frameworks and models as ISO/IEC 15939 and Goal-Question-Metric (GQM). Despite the recommendations from Chirinos et al. to use their measurement framework (MOSME), the ISO/IEC standard was more applicable due to its wider adoption in industry, the fact that it is standardized by an international standardization body and the easy coupling of theory and practice. The use of standardized view on measurement processes also provided the possibility for future benchmarking with other organizations (e.g. as indicated in [18]).

An alternative to the ISO/IEC 15939 is the GQM framework, although some differences exist. In particular, frameworks like GQM [20] are used to elicit a set of metrics which support decision processes in organizations, follow up on goals but not directly to monitor entities. The focus of measurement systems, however, is to support long-term monitoring of entities rather than to attain a specific goal. Software metrics need to be introduced, collected, and used during setting up of measurement programs (also called metric programs, which usually involves creating multiple measurement systems). To our best knowledge there exist a few frames of reference for assessing such programs. Jeffery and Berry [21] developed a framework for assessing whether a metric program

has chances of being successfully adopted. Several of their factors are related to such aspects as wide understanding and adoption of metrics, ease of use or flexibility of measurement systems. When designing our framework we addressed these issues. When assessing the success we combined Jeffery and Berry's framework with guidelines of Hall and Fenton [22] who presented a study of two companies, one of them being in the embedded software development, alike Ericsson. We also used the questionnaire developed by Jeffery and Berry to assess the potential for success of the metric program in our research project.

Factors contributing to successes and failures of metric programs are discussed by Fenton and Neil [23], who identify such fundamental problems as irrelevance of academically developed metrics for industrial use. Our approach is based on the identification of a stakeholder in the organization who is used as the reference point for assessing the usability of the developed measurement system (using the framework) – in other words, the measurements are developed with a particular use in mind rather than starting from theoretical considerations.

2.3. Similar measurement systems

The concept of a measurement system is not new in engineering or in software engineering – measurement instruments and systems are one of the cornerstones of engineering. In software engineering, we are used to working with metric tools rather than measurement systems. The difference is that metric tools and measurement instruments seem to be very similar, but metric tools and measurement systems are not. Measurement instruments (in other engineering disciplines) are suited for single purposes and usually collect one metric (e.g. voltage) whereas metric tools collect usually a number of metrics at the same time (e.g. length of the program, its complexity). Our framework is placed on top of metric tools with the focus on presenting calculating and presenting indicators rather than collecting metrics and is intended to be composed of multiple measurement instruments (metric tools). Other examples of measurement systems built in the same principles are:

- A measurement system presented by Wisell [24]: where the concept of using multiple measurement instruments to define a measurement system is also used.
- Computerized measurement systems in other disciplines facilitating the concept of measuring instruments, as presented in the following papers: [19,25–31]. All these measurement systems are (i) using the concept of measurement instruments, (ii) used in established engineering fields or physics, (iii) focused on monitoring current value of an attribute (status in our case) not on collecting metrics. Although differing in domains of applications these measurement systems show that concepts which we adopt from the international standards (like [32]) are successfully used in other engineering disciplines.
- Lowler and Kitchenham [10] present a generic way of modeling measures and building more advanced measures from less complex ones. Their work is linked to the TychoMetric [33] tool. The tool is a very powerful measurement system framework, which has many advanced features not present in our framework (e.g. advanced ways of combining metrics). TychoMetric provides a possibility of setting up advanced and distributed (over several computers) filters and queries for multiple data sources as it is intended to cover all (or at least very many) kinds of metrics and projects. The configuration of a set-up of one measurement system is much more difficult than developing a custom measurement system using our framework. The plethora of features and the fact that we intended our framework to be simple, easy to use and interpret rendered the TychoMetric tool too advanced for building measurement systems for wide spread throughout our organization. A similar approach to the TychoMetric's way of using metrics was presented by Garcia et al. [34]. Despite its complexity, the TychoMetric tool can be seen as an alternative as this tool can be used for more generic purposes not related to software metrics (e.g. advanced data presentation and advanced statistical analysis over time).

We also investigated metric tools used in software industry to compare the differences between these measurement instruments in other disciplines – in particular we studied publications on:

- Using video recordings in the process of measurements of fluids (hydrology), where the problem of "digitalizing" information from a physical world was discussed [35]. This problem is often encountered in software development organizations when using resource metrics.
- Network load measurements for a specific protocol using a dedicated measurement system (computer science), where we studied how a software measurement system is constructed and integrated with general software [30].
- Auto-calibration of measurement systems using feedback loops, where we studied the possibility of building-in auto-calibration mechanisms in our framework [36].

The above measurement systems used elements important for the measurement systems – auto-calibration, use of measurement instruments, and use of software components to increase flexibility of measurement systems.

3. Organizational context

In this section we briefly characterize the organizational context of working with metrics at the studied development unit within Ericsson (which we refer to as "organization" in the paper). The development unit develops software for a number of telecommunication network products. The software development projects executed in the development unit are usually large (100–200 persons) and span over long time – around 1 year.

3.1. History of metrics

The history of measuring at the studied organization started around 2002 when the need for quantitative data for decision support arose. In 2002, the awareness and the need for controlled metric program came into existence. The discussions continued for a number of years, including prototypes and early pitfalls. During 2003, the organization built various kinds of prototype metric programs including presentations for decision makers, which resulted in the development of common performance reporting in 2004. Since the awareness of measuring during these years was still being built-up, different metrics were used for same purposes (e.g. man-hours used, man-hours actual vs. planned, and use of time buffer were three metrics used to monitor cost) and same metrics for different purposes (e.g. using number of defects as a measure of product quality and test effectiveness) in the development unit. During 2005, an effort was undertaken to systematize the metrics and reduce the number of metrics collected. In 2006 an effort was established to build a single metric program for the whole development unit to control and monitor projects, products, and resources.

Over the years, the metrics used changed, although several are still valid (e.g. the in-service-performance, which is a measure of system availability with the criteria that the system should be fully operational 99.999% of the time, [37]). In this development unit, projects have been executed according to either RUP (Rational Unified Process, [38]), lean development or their variations.

Projects have been managed using the PROPS method [39]. The size of the studied development organization is several hundred persons working in several medium and large projects.

3.2. Types of metrics collected

The quality managers within the development unit work with quality-related metrics, for example number of defects reported, and test progress. These metrics are usually specified in quality assurance plans for each project, collected during projects and presented at project status meetings and daily/on-demand (for both the management team and the project management team). The metrics in quality assurance plans were collected in order to monitor project and product quality. And, since the maturity of the organization grew over time, the metrics evolved. At the beginning (around 2002), there were around 20 different areas, containing between 70 and 100 metrics. Some of the areas were intended to monitor whether a project adheres to process descriptions (e.g. measuring different activities in the projects). The metrics were collected manually and then used in performance reports (which were created manually as well). Nowadays, as the organization is much more mature, many of these metrics are not necessary – processes are followed and there is no need to spend resources on controlling that. Due to this, and similar factors, currently the organization collects metrics in seven areas, and the number of metrics decreased to approximately 20 metrics that monitor key aspects of projects. These metrics are updated automatically every day and always show the current status of the project. The measurement systems are available via web sites and there is no need for manual performance reporting of these metrics. As the competence in the company w.r.t. metrics collection and analysis is significantly higher than before, the demands for measurement systems grow constantly and there is a need to make the knowledge available to more persons than just a few individuals (as it was at the beginning).

The quality managers also work with high level project metrics supporting project managers (e.g. budget follow up, requirement coverage, time) and metrics for product managers (e.g. product performance after release). Moreover, the number of metrics collected using ISO/IEC 15939 based measurement systems is constantly growing. A growing number of stakeholders realized the benefits of having automated and daily updated information very easily accessible, which makes the number of measurement systems and metrics grow. Using the framework presented in this paper makes the development of measurement systems easier, faster and helps to satisfy the growing demands for measurement systems.

One of the factors facilitating automated metric collection and calculation is easy access to different databases and files. Metrics collected in the development unit are stored at such databases/files as:

- Requirement databases.
- Defect reporting databases.
- Log files.
- Databases storing information about the resources and economy – e.g. budgets, resource allocation, vacation plans.
- MS Excel worksheets with source information, for example: product downtime per unit of time, project progress, and subproject status information.
- MS PowerPoint presentations with specifications of targets and organization goals.

There is also a deal of information which can only be obtained from experts (e.g. predictions, estimates). This information needs to be manually inputted to the measurement systems. The variety of data sources makes it hard to obtain an overall picture of the metrics collected and to present them in a simple way that is quickly understandable by the stakeholders. Therefore there is a need for graphical specifications of metrics used in particular measurement systems.

3.3. Current goals and stakeholders

One of the goals of improving the work with metrics, which this study is a part of, was to provide a set of reusable metrics that could be used to compose indicators for various stakeholders. As the organization collects metrics related to products, projects, and resources; the number of metrics is quite large and there are multiple stakeholders who monitor and control the situation in the organization from different perspectives. Their information needs require presenting the data in different ways and calculating metrics differently. An example of multiple stakeholders for similar information is controlling the ISP (in-service-performance [37]) of the products in operation developed by the organization:

- The product managers have the information presented per product, which means that the information is aggregated from the information of ISP per release for all releases of the monitored product.
- The line managers have the information aggregated per organization – i.e. the ISP is aggregated from all releases of several relevant products (not all products are developed by the corresponding organization so the line managers do not need to control those that are not, unlike product managers).

The ISP of a particular node is always measured and stored in the same way to allow for comparison in the future and across development units within Ericsson.

4. Case study

This section introduces the design of the case study and outlines the execution of the study.

4.1. Design of the case study

Our goal was to study the way in which metrics are defined and collected in the studied organization. The objective was to improve the above processes and decrease the amount of manual labor for building, using, and maintaining information products. Therefore, our research question was:

> *How to simplify using and building measurement systems at Ericsson?*

In order to address this research question we conducted an action research with a flexible design [40] at one of the units at Ericsson. We decided to use flexible design for the case study as defined by Robson [40] as we lacked fully detailed information about the needs for introducing the metric program in the organization. In order to address this overall goal we devised a plan, which was revised regularly as new facts were encountered. The final version of the plan is presented in Table 1 as a sequence of steps.

Steps 3 and 4 were repeated several times until the framework was approved by the stakeholders and contained sufficient amount of features to improve their work.

The study was a part of an action research project. The roles of the three authors in the action research were:

- Practitioner 1 was the initiator of this research project since the part of the work of this practitioner was to improve measure-

Table 1
Research plan.

Step	Objective(s)	Research method(s)	Expected result(s)
1	(i) to understand the organization context (ii) to understand dependencies between processes and metrics in the organization (iii) to collect requirements for measurement systems	Qualitative analysis of interviews with: • line manager • quality managers • designer of measurement systems Statistical methods: development of prediction models (regression) for defect inflow prediction and their evaluation in projects.	History of metric initiatives in the organization; dependencies between metrics; requirements for measurement systems
2	(i) to identify relevant standards for measurement processes	Qualitative analysis: literature studies and informal interviews with managers within organization.	A set of international and internal standards specifying which metrics should be collected and how metric processes should be integrated in the organization.
3	(i) to create a framework for efficient development of measurement systems	Iterative development with prototyping.	A framework for developing measurement systems; measurement systems created by instantiating the framework
4	(i) to evaluate the framework in industrial environment (ii) to evaluate which factors might have contributed to the success of the metric program (in addition to the framework)	Qualitative analysis: interviews and workshop with • line manager • quality managers • designer of measurement systems • project manager Design and implementation of several measurement systems and their evaluation via interviews.	Improvement issues for the framework; aspects important when introducing metric programs into the organization and the role of the framework in these aspects.

ment processes in the development unit. His role in this action research study was to develop, implement, and evaluate measurement systems and introduce them into the organization and to state requirements for our framework. He had over 5 years experience as quality manager at Ericsson. This practitioner also had previous experience (over 5 years) in a similar position in automotive domain.

- Practitioner 2 was working with introducing metrics into projects and organizations for over 5 years. His role was to provide feedback about the good practices of working with metrics at the company, evaluate, and assess the framework. He had over 20 years of experience at Ericsson as a system designer, software process manager, line manager and quality manager.
- Researcher was working as a guest researcher at Ericsson (2 years) in the area of software metrics. His role was also to improve the measurement processes in the organization, in particular to identify metric dependencies, relationships and to optimize the number of metrics collected in the organization. His responsibilities in this study were to develop the framework, evaluate it and instantiate it to develop several measurement systems.

4.2. Execution of the case study

The steps 1–5 in our research process were conducted over a period of 1.5 years. The identification of the needs (step 1) of the company started in August 2006 and included developing prediction models for defect inflow in projects [41–43]. These models were linear equations for predicting defect inflow on weekly and monthly basis. We used multivariate linear regression and principal component analysis to create these models. The particular reason for working with prediction models was to understand the dependencies between processes and metrics in the company (i.e. to achieve objective ii for this step). We introduced these models into the organization, which resulted in improving accuracy of predicting the number of defects which need to be addressed before the software is released. In order to develop these models we performed statistical analyses but also conducted interviews with project and quality managers in order to identify factors important when making predictions and evaluating usefulness of the resulting models.

The interviews with managers and designers were conducted at several occasions with the goal to obtain their opinion on the requirements for measurement systems (step 1, objectives (i) and (iii)). Since we followed iterative development with prototyping, similar interviews were repeated regularly (after each prototype was ready) during the whole duration of the study.

Step 2 was conducted during a 3 month period during fall 2006. We performed literature searches and studies both in internal Ericsson databases and external ones (e.g. ISO/IEC, IEEE, ACM, Springer-Link, Science Direct). We also asked experts within Ericsson about the standards they might find applicable.

Since January 2007 several measurement systems were built by the unit and introduced into use (step 3). These measurement systems were structured according to the ISO/IEC 15939:2002 standard, in particular they presented information in the form of indicators. One of the observations was that the development of a measurement system was an effort intensive task and involved quite a deal of script programming.

At the same time we observed a growing demand for measurement systems at the organization and the company. With this respect the process of building measurement systems was time consuming and was one of the blocking factors for wide spreading of measurement programs in the organization. By using the framework together with a set of pre-defined metrics the needs for new measurement systems could be satisfied as smaller programming effort allowing more persons to develop and maintain measurement systems.

The framework for the measurement system was developed in December 2007 and was intended to (i) decrease the amount of time required to develop a measurement system, and (ii) remove the requirement for script programming (in case a metric was available or could be imported from another measurement system). At that time a new version of the standard was released: ISO/IEC 15939:2007. The new version did not introduce changes in the parts used in our research. The framework was developed by the guest researcher and was introduced into use by the practitioners.

5. Results

In this section we present the measurement framework, how it is used to build measurement systems and its evaluation in the projects. The principles that govern the structure of measurement systems are presented in Section 5.1. These principles influence how the framework is used (i.e. how the measurement systems are being built), which is described later in this section. A glossary of terms used in this section from ISO/IEC 15939:2007 (e.g. base/derived measures, indicators) can be found in Appendix A.

5.1. Requirements for framework

The requirements for the framework were elicited during the whole duration of the study. In this section we briefly present the final set of high-level requirements, which the framework fulfills (validation was done together with the designers of measurement systems and stakeholders):

1. The framework should realize the information model from the ISO/IEC 15939 standard.
2. The framework should be based on MS Excel and other MS Office software.
3. The framework should be able to collect the data from existing tools in the company (the list of tools was provided).
4. The framework should provide easy ways of defining, updating, and removing metrics and indicators (including decision criteria).
5. The framework should provide the possibility to programmatically modify measurement systems (e.g. by using automated tools to add/remove metrics without manually changing the source code).
6. The framework should enable building modular measurement systems (i.e. separation of base, derived measures and indicators).
7. The framework should enable run-time importing/exporting of values of metrics between separate measurement systems; metrics should not be redefined at several places in order to minimize the risk of inconsistent definitions.
8. The framework should support defining metrics compatible with quality metrics defined in ISO/IEC 9126 [44] (which was found to be important for the development unit).
9. The framework should support the notion of measurement instrument as defined in standard ISO/IEC Vocabulary in Metrology [32].

From the non-functional requirements, usability of the framework, reliability of resulting measurement systems, and maintainability were of premiere importance. It should be very simple to set-up and maintain measurement systems using this framework (otherwise we risked that the framework was not adopted by the designers of measurement systems or the resulting measurement systems not being adopted by the stakeholders).

5.2. Framework for building measurement systems

The framework for building measurement systems which we developed in our research is a predefined MS Excel workbook (realizing requirement 2) which contains:

- API for measurement systems in form of Visual Basic for Application modules with functions and procedures for defining and using the metrics (providing the possibility of programmatically modifying measurement systems at run-time, realizing requirement 5).
- Built-in logging component (which uses the existing logging component available at the company).
- Wizards supporting the development of measurement systems, e.g. adding indicators, building derived measures, updating values of base measures (realizing requirement 4). A *base measure* is a metric which measures a single attribute of a single entity – for example number of defects found in a component. A *derived measure* is a metric which combines base measures – e.g. average number of defects per component (combination of number of defect and number of components). We choose to keep the original terms base measure and derived measure as defined in the ISO/IEC standard rather than changing them to base/derived metrics to be consistent with the vocabulary of the standard.

The API is intended to provide the possibility of automatic reconfiguration of measurement systems during their execution (e.g. to dynamically change decision criteria for indicators, obtain values of metrics and store them in a measurement database).

5.2.1. Architecture of the framework

The framework allows building measurement systems of a specific architecture, which is presented in Fig. 1. The arrows represent data flow.

The measurement systems are built on top of MS Excel in order to reuse the toolkit that is already used to collect metrics and data in the company. The measurement system contains three main components, corresponding to the elements in the measurement information model specified in the ISO/IEC 15939:2007 standard: indicators, derived measures (DM), and base measures (BM). The dotted line around these components is intended to show that these three components might exist in one or several execution spaces – i.e. being separate MS Excel instances (any combination is allowed, e.g. BM and DM in one file whereas indicators in another one). In the case of using several instances, the metrics are imported when needed, e.g. base measures are imported when needed for calculation of derived measures. The possibility to split

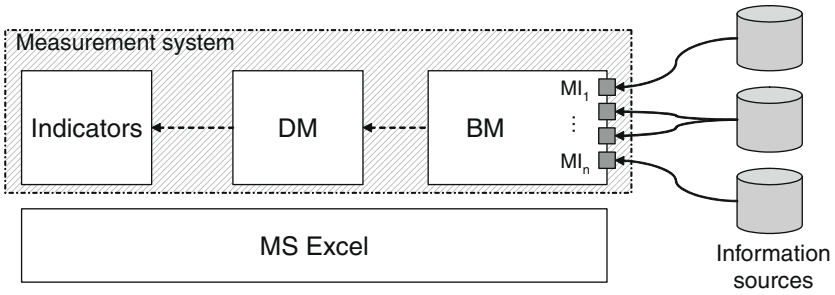

Fig. 1. Architecture of measurement systems.

different parts in different files was one of the requirements for the framework.

The component with base measures contains the possibility of defining additional scripts that can obtain data from databases – measurement instruments (MI), since these scripts are used to assign values to metrics (i.e. to measure). In cases when measuring is done manually, the measurement instruments are outside the scope of the physical architecture of the measurement system. Although the manual measurements are usually supported by the use of checklists and documents describing the measurement processes, we distinguish here between the automated measurement process with the use of measurement instruments and the manual process using other means. The measurement system naturally supports the manual process since it provides wizards for assigning values to metrics.

Since MS Excel has the possibility of using COM (Component Object Model) components, the measurement instruments can measure values in the following way:

- open and import values from other MS Excel workbooks,
- connect to databases via their COM API,
- open text files, convert them to MS Excel format and import values from these (provided that the text files are of a specific structure).

The complexity of such scripts depends naturally on the complexity of the measurement process and the need for reconfiguring the data. The reconfiguration might be a complex task and might require extrapolations. A typical example is the reconfiguration of data presented by week into data presented by month, which might require assumptions about the mapping since months and weeks do not match (each month has a non-integer number of weeks). Automated reconfiguration of data reduces the amount of manual work when working with software metrics.

As the measurement process should not alter the measured object, the development of the measurement instruments does not require changes in the structure of information available in the organization. The measurement systems built by instantiating the framework can be integrated with the existing tools in the organization.

5.2.2. APIs of the framework

Each of the components in the measurement system has a corresponding API (Ind_API, DM_API, and BM_API) to support the requirement for adding/removing/updating metrics programmatically. The logical view of the architecture is shown in Fig. 2 (the exact parameters are omitted for clarity of presentation). In addition to the above APIs, the system provides also API for automatic updates of the measurement system. The updating API is used to update the measurement systems during a specified time (or on demand) from external tools that support COM interfacing.

Each of the interfaces on the right-hand side is used to add/remove/update metrics in the measurement system. The manipulation of metrics (i.e. measuring and obtaining the values) is done by the procedures and functions that have a suffix "Value", e.g. "updateBaseMeasureValue()" which updates the value of a given base measure, whereas "updateBaseMeasure()" updates the definition of the metric itself. Attaching scripts, which are measurement instruments, is done via a simple subscriber–publisher mechanism. The mechanism invokes registered scripts that in their turn

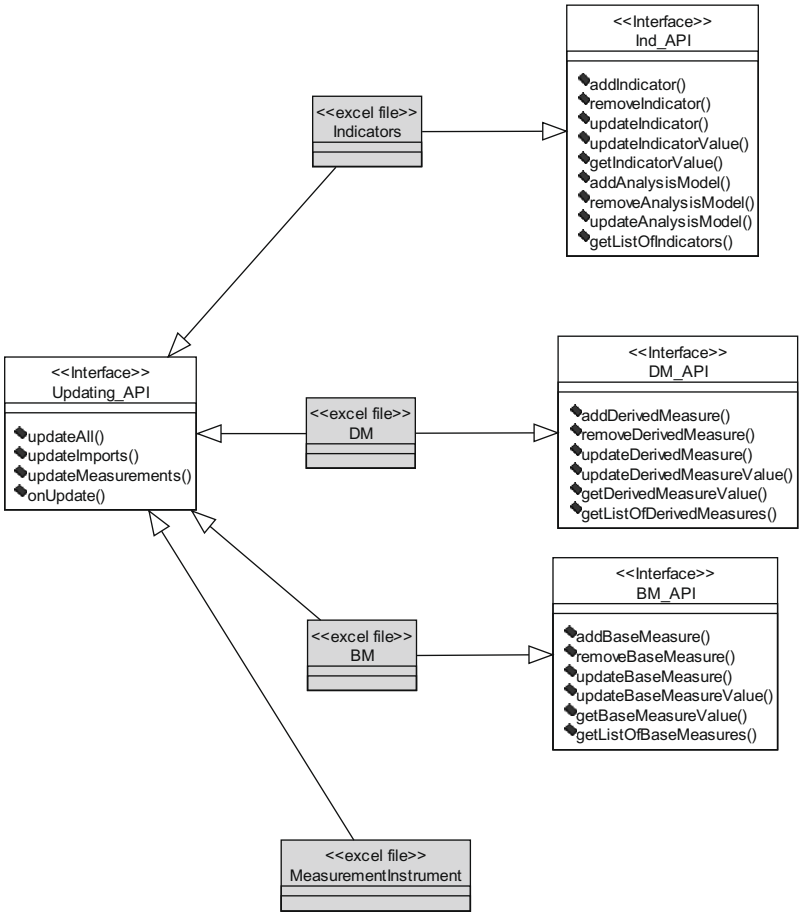

Fig. 2. Logical view of APIs in measurement systems.

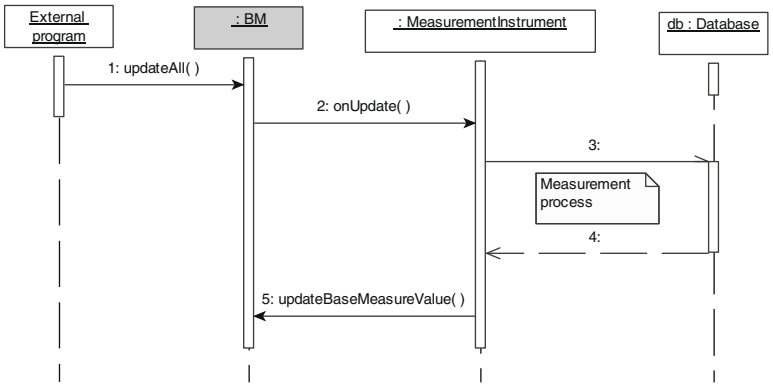

Fig. 3. Using a measurement instrument when updating the measurement system.

measure an object's attribute and invoke "updateBaseMeasureValue()" to update the value of the metric. A typical scenario when an external program (or a person by using a menu option) requests the update of the measurement system is presented in Fig. 3.

Fig. 3 shows a scenario (as a message sequence chart/sequence diagram) when an external program (e.g. a task scheduled to collect the data during nighttime) updates the value of a base measure. The process starts when the update process invokes the updateAll() operation for a component which contains base measures (:BM). The component invokes the operation onUpdate() of the measurement instrument (:MeasurementInstrument), which reads the values from the database (db: Database). As measurement processes are specific for various kind of databases, attributes being measured, etc. this part is represented as a note in this diagram. After the information is obtained from the database (i.e. the measurement process has been successfully finished) the measurement instrument updates the value of the base measure by invoking the operation updateBaseMeasureValue().

As shown in the diagram in Fig. 3, it is the measurement instrument that is used to perform the measurement process (in this scenario interacting with, and fetching information from, a database). The data source of the information – the database – is an external entity (thus not part of the architecture view in Fig. 2) and since the interaction with the databases depends on the measured attributes, no generic way has been developed so far.

5.2.3. Measurement instruments

Measurement instruments play an important role as they separate the measurement system (i.e. the information model from ISO/IEC 15939) from the process of measurement. The role of measurement instruments is to transform an attribute of an entity to its numerical representation and store it as the base measure. This separation of ISO/IEC 15939 from physical measurement processes addresses the requirement for modularization of measurement systems and allows the measurement instruments be reused (to some extent). The way in which measurement instrument work can best be presented in the following example.

A measurement instrument for measuring number of requirements for a project is presented in Fig. 4. This measurement instrument measures the number of functional requirements (denoted as

```
1     Public Function measureNumberOfRequirements(strProjectName As String) As Integer
2         Set RQDB = CreateObject("requirements_db")
3         For Each SelectedProj In RQDB.Projects
4             Set oProj = RQDB.OpenProject(SelectedProj.FullPath)
5             If oProj.Name = strProjectName Then
6                 If Not oProj Is Nothing Then
7                     For Each rqType In oProj.ReqTypes
8                         If (rqType.ReqPrefix = "FR") Then
9                             Set reqs = oProj.GetRequirements(rqType.ReqPrefix, 2)
10                        End If
11                    Next
12                End If
13            End If
14        Next
15        measureNumberOfRequirements = reqs.Count
16    End Function
```

Fig. 4. Measurement instrument code for counting number of requirements in requirements database.

"FR" in Line 8) in a given project (denoted as "strProjectName" in Line 1). The data is stored in a requirements database (for confidentiality reasons the vendor cannot be provided). It is written in Visual Basic for Applications (VBA), which is a component of Microsoft Excel.

The function starts with connecting to the requirement database (symbolically denoted as "requirements_db" in Line 2) and iterating over all projects (Line 3), opening the project (Line 4), and checking if the project is the desired one (Line 5). If the project is the one which is measured (Line 5) then the code searches for the functional requirements (Line 8) and retrieves the collection of all functional requirements (Line 9). The measurement is performed in Line 15, which is also the line where the value is returned. The code corresponds to steps 3–4 in sequence diagram presented in Fig. 3, denoted as "measurement process" and presents the process of quantifying an attribute (number of requirements) of an entity (project).

This measurement instrument is used to measure a specific object – in this case a specific project. An example code which uses this measurement instrument to measure the object "MyProject" is presented in Fig. 5.

The code using this measurement instrument sets the objects to be measured (Line 2) and stores the result of the measurement (Line 3).

5.2.4. Wizards and error logging

In order to successfully develop measurement systems the framework contains wizards which guide the designer in creation of the measurement system and later the stakeholders in updating measurement systems (if needed). The wizards only provide a graphical user interface for the API so that the developers and stakeholders of the measurement systems need not to resolve to programming to create a measurement system. The wizards significantly increase the usability of the framework and provide the stakeholders with the possibility to customize their measurement systems.

In order to be able to notify the stakeholders and designers when an automated measurement process is not successful, the measurement framework has a built in logging subsystem. It uses existing error logging infrastructure at the studied organization as that was one of the requirements (requirement 3). The measurement systems are intended to be run automatically (usually during night time) so they have to be fail-safe, i.e. they cannot crash the servers where they run or start infinite loops. The error handling procedures prevent such situations and use the logging subsystem to leave traces of problems with measurement during the updates of measurements. The error reporting also notifies a dedicated person via e-mail that errors have occurred.

5.3. Developing measurement systems

This section presents the process of developing a measurement system in general. An example measurement system to illustrate the point is shown in the evaluation section (Section 5.4).

When developing measurement systems we need to specify metrics used, measurement instruments, and measured entities. These specifications are presented in Fig. 6.

At the initial stage one needs to describe which entities are to be measured and which metrics are to be collected, as it is presented as the top plain of Fig. 6. This specification is a model that describes metrics (based and derived) and indicators for specific entities, but it does not specify which objects are measured. In this

```
1     Public Sub onUpdate()
2         iNoOfRequirements = measureNumberOfRequirements("MyProject")
3         Call updateValue("Number_of_requirements", _
                           iNoOfRequirements, _
                           ActiveWorkbook)
4     End Sub
```

Fig. 5. Using the measurement instrument.

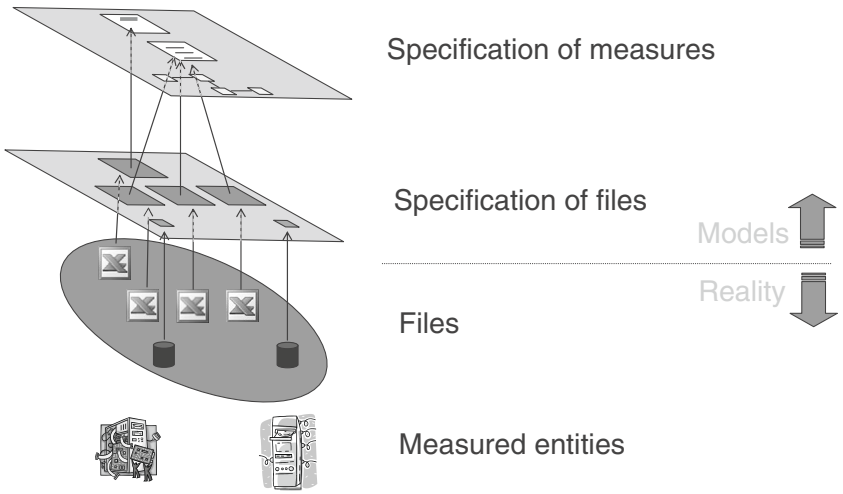

Fig. 6. Dependencies between specifications of measures, specification of files, files and measured entities.

sense, the top level is the model of metrics and measured entities. The rectangular planes mean that these elements are specifications, whereas the oval plane means that these elements are real-world objects – files – which constitute a measurement system.

This specification ("model") is instantiated for each information product, which is shown in the specification of files. This specification is a model of how metrics are instantiated – i.e. metrics collected and files used. Each file in this model corresponds to exactly one element from the specification of metrics. The specification/model of files corresponds one-to-one with the actual files which contain actual metric values (which are shown as "Files" in the figure). The scripts in the files connect to the databases and collect data (in other words measure) from real products. The databases, on the other hand contain data collected from objects (e.g. nodes in operation in a telecom network, or an on-going project).

The goal of having the specification of metrics and specification of files separated is that the metrics are to be reused for measuring different objects. The reuse is done by making a copy of the file with base measures, and adjusting the objects that are measured (i.e. applying the same measurement instruments on different objects). The goal of the organization is to have (as time goes by and more measurement systems are built) a repository of reusable instruments, which can be used to build new measurement systems for new purposes.

An important aspect of the use of the framework is the ability of reusing metrics via importing. This provides the possibility of creating a metrics repository and thus to reuse metrics as Lego bricks to build different measurement systems. The exact calculations naturally have to adhere to the state-of-the-art in measurement theory (e.g. combining values, admissible transformations).

5.4. Examples of measurement systems

The studied development unit has created and uses a number of measurement systems, for example:

- Measuring ISP (as presented in the example in this paper – Table 2) for the manager of the product management organization; example metrics in this measurement system are:
  - Product downtime per month in minutes.
  - Number of nodes in operation.
- Measuring project status and progress – for project managers who need to have daily updated information about requirements coverage in the project, test progress, costs, etc.; example metrics in this measurement system are:
  - Number of work packages finished during the current week.
  - Number of work packages planned to be finished during the current week.
  - Number of test cases executed during the current week.
  - Cost of the project up till the current date.
  - Budgeted cost of the project up to the current time.
- Measuring post-release defect inflow – for product managers who need to have weekly and monthly reports about the number of defects reported from external customers; examples of metrics:
  - Number of defects reported from field operation of a product during the last month.
  - Number of nodes in operation last month.
  - Number of nodes which reported defects.
- Summarizing status from several projects – for department manager who needs to have an overview of the status of all projects conducted in the organization, e.g. number of projects with all indicators "green".

These measurement systems were instantiated for a number of projects and products (depending on the measured entities). Each of these instances has a distinct individual as stakeholder (who has the role of project manager, product manager, etc.) who uses the measurement system regularly.

Metrics used in our measurement systems can both be collected automatically from databases or manually from persons when the data is not stored in databases (e.g. by asking the project manager how many designers are assigned to remove defects from the software in a particular week). The sources of information are defined in the metrics specification and the infrastructure specification for the particular measurement systems.

6. Demonstration of using the framework

As an example how to use the framework we present a measurement system constructed to measure ISP (in-service-

Table 2
Specification of the measurement system for monitoring In-Service-Performance (ISP).

Stakeholder	Product manager
Information need	Are we fulfilling the X_1 availability promise?
Indicator	Unplanned downtime for all products
Analysis model (decision criteria)	Green: X_1 to 100% availability Yellow: X_2 to X_1 availability Red: below X_2 availability
Derived measures	**PA**: Product availability last month extrapolated for one year [minutes]
Measurement Function	**PA** = sum(**RA** for all releases)/**NR**
Base measures	Object: Release **RA**: Release availability last month extrapolated for one year [minutes/year] Object: Product **NR**: Number of releases [number]
Measurement method	For **RA**: 1. Collect downtime logs from all nodes running the release for the last month 2. Calculate average downtime for all nodes running the release 3. Count the number of minutes in last month 4. RA = number of minutes in last month – average downtime for all nodes running the release For **NR**: 1. Count the Number of Releases of software which are in operations

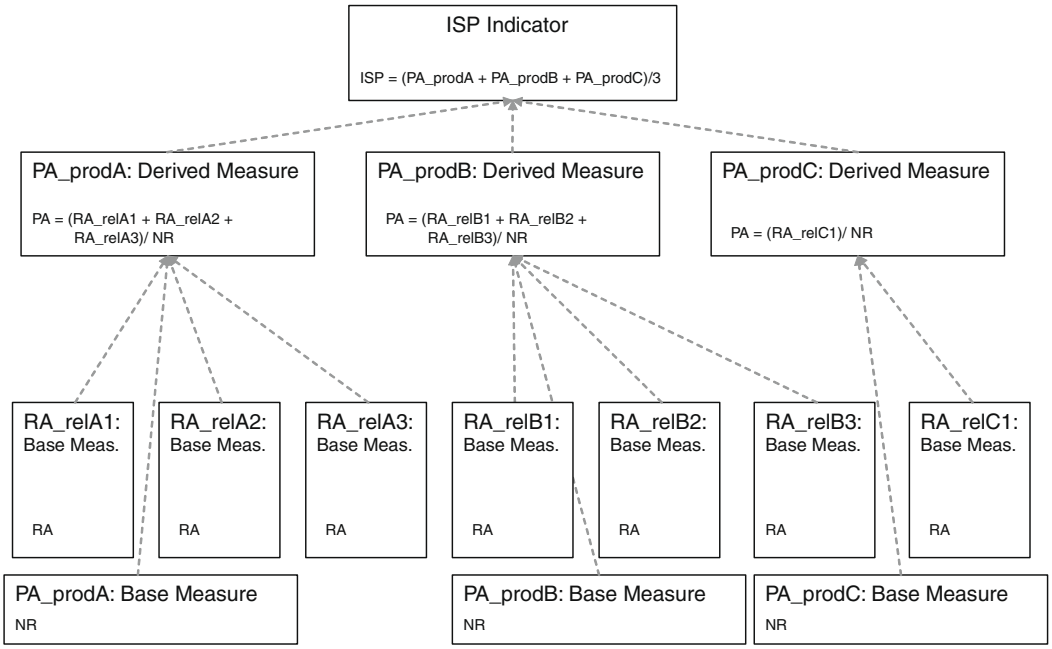

Fig. 7. Instantiation of measurement system for ISP; PA, NR, and RA are defined in Table 1.

performance) of telecom network nodes in operation. The measurement system is defined in Table 2 and Fig. 7; the values in the analysis model are not given due to confidentiality agreement with Ericsson.

As the studied organization has developed several releases of the software for the nodes, the organization collects the information per each release for each product. In total the organization monitors three products (Product A, Product B, and Product C) so there are three instances of derived measures. Each product has a number of releases in operation (the releases which are no-longer maintained, out-dated are excluded): Product A – three releases in operation; Product B – three releases in operation; Product C – one release in operation. Therefore there are multiple instantiations of base measures for each product (Product A – three instances of base measures, Product B – three instances of base measures, and Product C – one instance of the base measure). Therefore, the specification is instantiated as presented in Fig. 7.

In practice, each rectangle in Fig. 7 is a separate file in this example (although there is no requirement that it should be a separate file). Each file measuring a release is identical in structure, but differs w.r.t. which object is used as input for the measurement instrument (i.e. measurement systems are instantiated for different products/releases). The measurement instruments are the same (i.e. they realize the same algorithm/measurement method). The files with derived measures are also identical in structure for all products, but they differ in the values they import. Finally, there is only one file with the indicator that satisfies the information need of the stakeholder. The positioning of the specification and the files are presented in Fig. 8.

The measurement system used at the company is presented in Fig. 9. The measurement system contains the total ISP indicator (the first indicator from the top), and several more detailed indicators, which use the same analysis model to present ISP indicators per product and per release. The values of the indicators (downtimes) have been hidden due to confidentiality of the data.

The measurement system updates the information from databases daily and updates the presented information accordingly.

7. Evaluation of the framework

After the framework was in its first release we instantiated it in a number of measurement systems and evaluated through workshops with stakeholders, a developer of other measurement systems, and quality managers.

7.1. Industrial evaluation

Our intention is to present the framework for constructing the measurement systems and its evaluation. The evaluation is done by:

- Using the framework to construct measurement systems which are used in the organization.
- Conduct interviews with stakeholders, developers of measurement systems and quality managers working with measurement processes in the organization, in particular:
  - Stakeholder 1: an open interview with a line manager with several years of experience in management and software development. The stakeholder was also involved in the development of a measurement system for him, during which he (in addition to defining indicators and metrics) contributed with feedback.
  - Stakeholder 2: an open interview with a project manager with several years of experience; this stakeholder was also involved in developing one of the first measurement systems where (as stakeholder 1) contributed with feedback.
  - Developer of other measurement systems: continuous feedback during the development from a senior quality manager with more than 10 years experience in the position; also an interview after the framework was used.
  - Quality managers: workshop with two senior quality managers with more than 10 years experience in software development and quality management; also interviews when they assessed the framework.

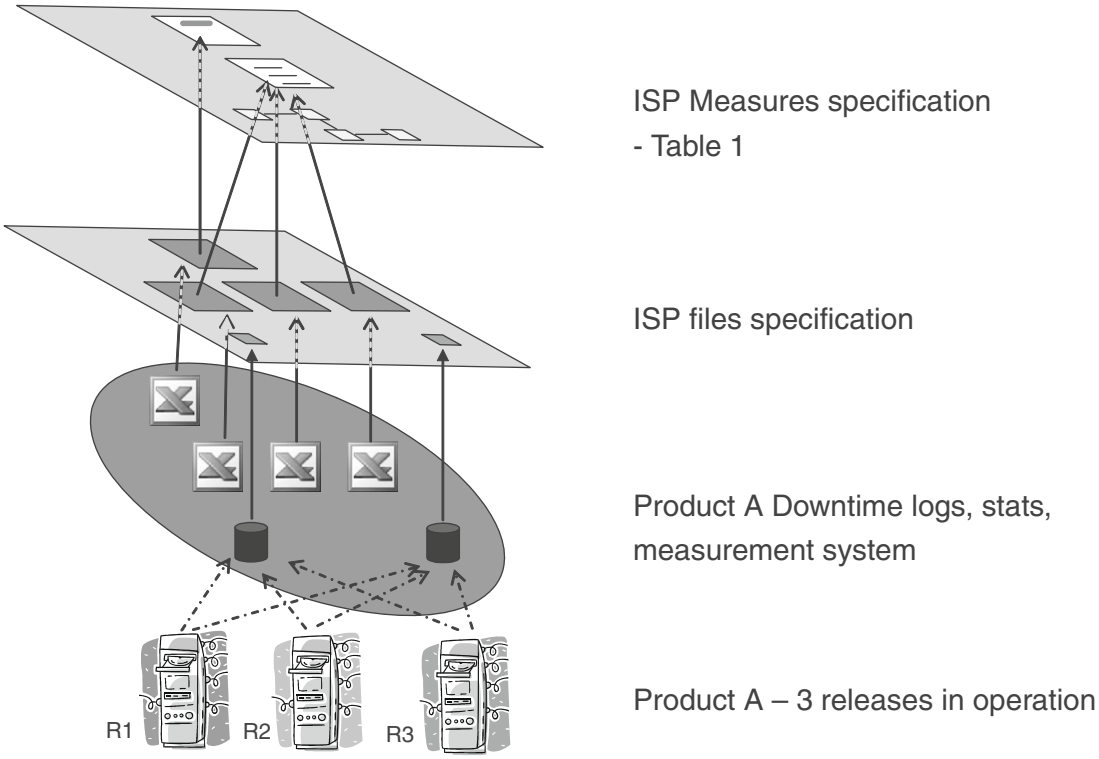

Fig. 8. Specifications and real world objects in measuring ISP. Solid lines show instantiation, dotted lines show reporting (information flow).

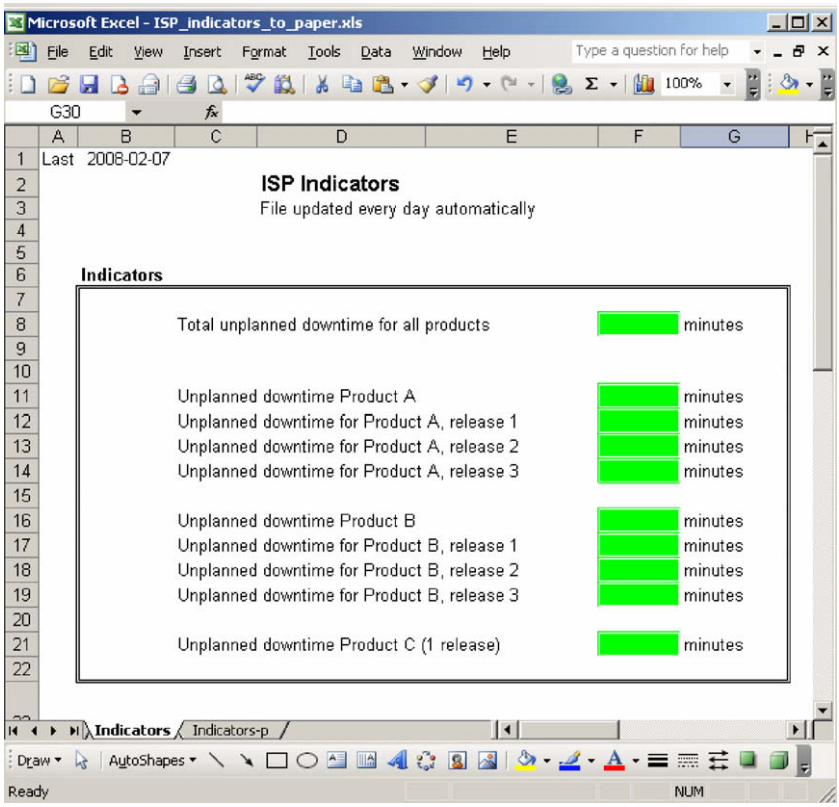

Fig. 9. Measurement system for ISP: presentation of indicators.

The goal of the interviews was to obtain the feedback from the practitioners and their subjective opinion on the usability of the framework and this way of constructing measurement systems.

Re-development of an existing measurement system for project monitoring showed a decrease of the amount of Visual Basic for Application code by ca. 35%. The remaining code was required

for the development of measurement instruments as no metrics could be reused when building this measurement system. The re-development of the existing measurement system also provided us with the possibility of testing whether the framework works correctly (verify the framework). The two measurement systems were working alongside each other and the results were automatically compared on a daily basis (after the measurement systems were updated each day, the information was compared). This verification did not find any errors in the framework or in the re-developed measurement system.

The framework provides the stakeholders with the possibility to add/remove/change metrics and indicators, which is an important feature given the fact that metrics and indicators change over time as projects or organizations change (which happens often in large organizations, including Ericsson, as described in Section 3).

7.2. Spread in the development unit

One of the measures of success of our framework was the spread of measurement systems in the development unit, which size is several hundred persons. Since the ISO/IEC 15939 was introduced and the culture has changed from measuring to fulfilling information needs, metrics became more demanded. In particular the following happened in the development unit as a result of our research:

- Metrics in Balance-Score-Card (BSC) used by top management team are collected automatically, which saves time and gives the management team always up-to-date status:
  - As a result of that the management team asked for more metrics fulfilling new information needs – the situation changed from having no automated metrics to providing measurement systems for various aspects in several large software projects. Using the measurement system for BSC influences indirectly the whole development unit.
- The line organization now has more detailed metrics for performance reporting and monitoring due to the fact that metrics are "easy to find" and "regularly updated". The metrics collected are used to create custom reports requested by various managers; this made the work of quality managers more advanced – instead of administering metrics, quality managers focus on advanced statistical analysis of metrics and indicators. This in turn affected the efficiency of the work of quality managers – they could work with more projects and provide them with better support.
- Projects have more statistics – i.e. the statistics were improved by making them more advanced and more precise (more suited for answering the information need rather than summarizing data).
- Projects have indicators which improve their "steer on facts" principle; in addition the projects have more metrics collected/updated more often which leads to much better control.
  - One of the projects made significant adjustments in resource allocation and avoided problems due to the information provided by one of our measurement systems: after getting an early warning from the measurement system for predicting defect inflow (based on the methods presented in [43]) the project took measures, e.g. reallocating resources at an early stage of the project. Without these predictions the situation could escalate and the project could be over budget and delayed.
- Stakeholders, quality managers, etc. have easier access to historical data and are able to make better comparisons in shorter time.
- The measurement systems have impact on a significant number of employees in the development unit (which consist of several hundred persons) – either from the perspective of line managers, project managers, or product managers.
- The framework has improved the work of the quality managers working in the development unit – their work has been automated and therefore quality managers can monitor more projects and products on a daily basis or monitor more aspects.

Once measurement systems begun spreading in the organization we noticed that decision criteria were used as support for software process improvement. Generic indicators, for example defect inflow, were reused in different projects and some of the "new" projects set up their acceptable levels of defect inflow at lower levels, since they were aiming at being better than the past projects. The lowered thresholds for acceptable number of defects introduced more ambitious quality goals for the projects. Since exceeding the thresholds is treated seriously at the company, such an approach supports process improvement activities.

As the framework is used for several purposes within Ericsson, we believe that it is also usable for other companies. In the future, we intend to prepare a public version of the framework available for other companies through a web portal.

7.3. Feedback from interviews

Prior to the workshop where we evaluated the framework we conducted the interviews where we evaluated the complete metric program in the development unit using the framework of Jeffery and Berry [21]. The evaluation showed that the metric program was indeed very successful and satisfied the needs of stakeholders, quality managers, and developers of measurement systems. Jeffery and Berry categorize requirements important for success of a metric program into four categories (presented in Table 3) and assessing a metric program is done by assigning a value 0–3 to each requirement in each category where 0 – requirement is not fulfilled completely, 1 – the requirement is fulfilled to some extent, 2 – requirement is almost fully fulfilled, and 3 – requirement is fulfilled completely. The total score is an average of these answers, e.g. score 100% is when all requirements are fulfilled completely (in category "context" below this would be 300 points = 100%) whereas 0% is all requirements are not fulfilled completely. The assessment of fulfilling each requirement was a consensus between two quality managers, developer of measurement systems and a stakeholder.

Table 3 shows that almost all requirements were fulfilled by the metric program in the development unit. The input and product categories (which are mainly affected by the framework) are fulfilled to 80% and 76%, respectively. This indicates that the use of the framework and automated measurement systems were the key factors which contributed to this success (it was also confirmed by all involved in this evaluation). During the workshop with the quality managers they found this product to be useful in their work, and to fulfill their expectations.

Table 3
Summary of results from using Jeffery and Berry's requirements'; larger score – better.

Category	Number of questions	Score
Context	10	79%
Input	8	80%
Process	17	64%
Product	5	76%

We also obtained feedback that it was about 50% faster to develop a measurement system using the framework than in the previous way (which was an estimate from the developer of measurement systems). This improvement was caused by the fact that all the code which was generic for all measurement systems was already in place and tested, so the code to be written was only for measurement instruments, and even such code could be reused (as this code is related to the databases used in the development unit and the number of those is limited). The developers of the measurement system do not change the code of the framework, which implies that they cannot introduce new errors to this code. As opposed to the previous approach, where the reuse of code was at the lower-level (copying and changing parts of the source code between measurement system), this results in less error-prone measurement systems.

Stakeholder found the framework as an improvement of the way of working with metrics and providing possibilities of reuse of metrics. The main positive feedback of the stakeholder was that the framework was simple to use. Being very intuitive and easy to use allowed for the framework to spread throughout the organization. The stakeholders are usually very busy and adding to their responsibilities is not a feasible option. The stakeholder found the framework satisfying his usability requirements.

The developer of measurement systems concluded that the framework simplified his way of working significantly. Before the framework was ready, the developer was dependent on other people with more expertise in visual basic for applications (VBA) to write the code, and the process of developing and setting up a measurement system was very time consuming. Without a framework, only to develop a measurement system took approximately one week. Now, with the framework, this takes approximately one hour (for measurement systems of very similar complexity) and does not require being dependent on external expertise in VBA. The framework provides the stakeholders with the possibility of altering decision criteria, analysis models, base/derived measures, etc. without the need to alter the VBA code (as it is not their task to write computer programs). The no-programming made the stakeholders independent from the developer of the measurement system once the system was deployed.

The developer also used some concepts from the framework to improve the existing measurement systems – in particular the customizable decision criteria, or the presentation of metrics. This indicates that the concepts included in the framework were not specific to it, but rather applicable to measurement systems based on ISO/IEC 15939.

All interviewees found our framework as help in working in a more structural way – according to the standard and with more structured way of thinking. This allowed to spread the measurement systems to more people and to elevate their understanding what a measurement system is and do not measure in an ad hoc manner.

7.3.1. Factors facilitating growth of metrics culture

In order to find out which other factors could have contributed to our framework being successfully adopted in the organization, during the interviews we asked which factors contributed to the growth of metrics culture in the organization in general. By asking this question we wanted to assess which role the framework played in the spreading the metric culture.

We found that the framework contributed to the change of metric culture in the organization. As opposed to the past, software metrics are currently discussed at all levels in the organization and the need for using metrics for decision support is growing all the time. This growing need requires infrastructure where metrics can be used (e.g. combined, summarized) by end users (e.g. managers, system owners), in a short notice (which is supported by the framework).

In particular we found the following factors important for the wide-spread of the use of metrics (listed in descending order of importance):

- Cultural change to considering information needs rather than metrics – project, line, and product managers do not focus on finding the most important metrics, but on defining the information needs and indicators, which makes the process of identifying data to be collected (and processed) more efficient;
- Drivers of metrics in the company – when data is collected there is a dedicated person/role who presents the data and prevents it from being biased; that person is able to explain in detail how the data is collected, processed and presented to ensure that the numbers reflect the reality.

The above two factors and their combination is beyond doubt the two most important factors – there should be a need for the data and there should be a person (or a group of persons) who can provide this data. The usability of our framework contributed to the fact that these persons need not to be experts in programming, which makes it possible for managers to play this role. Additional factors which contributed to the above two factors being successful (or even possible) are:

- Automation of metrics collection – metrics are provided on a daily basis and require no manual work to collect them (which is a cost-effective solution);
- Providing means for assessing whether indicators can be trusted or not – for each indicator there is a number of checks which result in notification to the stakeholder if there was a problem with the data collection or processing;
- Providing the stakeholders with the possibility to customize, combine, or add/remove metrics and indicators in the measurement systems; changing the analysis model is one of the features which the stakeholders ask for (which was also one of the requirements for the framework).
- Ability to integrate the measurement systems with existing infrastructure – the measurement systems do not require introducing new tools for the elements being measured (e.g. no new reporting tools for software designers); this was also a requirement for the framework.
- Building in-house competence in software metrics – the organization invested in learning advanced ways of working with metrics rather than sub-contracting this work to a 3rd party consultancy company.

The above factors are identified and prioritized based on our experiences with setting up the metric program.

It should be noted that there are barriers when adopting measurement systems in large organizations. One of the barriers which we observed was the difficulty in changing the mind set, switching from talking about "metrics" to defining "information need". It is always easy to define a metric or two, but it requires some effort to define the information need that exactly pinpoints the essence of the matter. For instance it is easy to define the number of test cases being executed, but it is far more complex to define how this information is used to monitor progress of testing; e.g. should it be the number of executed test cases over the number of planned test cases or the number of passed test cases over the number of planned test cases (or even other combinations). Another barrier is the insecurity of trusting the data/status of the information need – the stakeholder has to know that the data is up to date and correct. So it is imperative for the developers of measurement systems that they build them up in such a way that the stakeholders feel confident with the information presented.

8. Threats to validity

We group the threats to validity of our case study according to the categories discussed in [45].

The main *external validity* threat is the sampling and generalization of results. We were only studying one development unit within one company. The framework, however, is not specific to the company or development unit, but structured according to two ISO/IEC standards, which increases the possibilities of its use in other contexts. As we constructed several measurement systems for various purposes, we observed that the framework can be used with different entities – e.g. projects, products, specifications, models, resources, plans. Successful spread of the framework in a large organization shows that the framework is not specific for a single project or purpose.

Our study was done over a period of several months (18) in a large development organization; hence we cannot claim the full *internal validity* of our claim that the framework was the only cause of the observed improvement of metric processes. The main threat is the maturity effect as defined in [45,46]. Since our case study was done over a period of 1.5 years, we certainly learned along the way and we influenced the environment around us in the development unit. However, as it was indicated in Section 7.3.1 when discussing the factors influencing the successful adoption of the metric program, automation and using indicators instead of metrics, were very important for the wide adoption of metrics in the company. As it was also stated by the stakeholders, managers, and developers of the measurement systems, our framework provided a solid help in their way-of-working with metrics in the organization.

A history effect [45] could also influence the results of the study; the history effect means that there is a danger that it was the background of the individuals in the organization that caused the observed improvements, not our framework. We believe that this threat is not valid since our framework uses the knowledge in the organization to improve the way of working. The framework introduced such concepts to the measurement systems as customizable decision criteria, new presentation of information and easy reuse of metrics. As described in the interviews these factors in particular indicate that the history effect did not have the influence on the observed improvement.

Since we only interviewed a few respondents and built measurement systems, we cannot use statistics to verify hypotheses on the improvements of way of working in the company. This threat to *conclusion validity* of our claims that the framework is indeed useful makes our evaluation an initial one. We plan to run experiments to compare our framework with other frameworks discussed in the related work section.

9. Contributions

The main contribution of our research is the new framework supporting the development of measurement systems according to the standard ISO/IEC 15939:2007 and its positive industrial evaluation. The important parts of the framework which make it successful are:

- the organization has the access to the framework which makes building measurement systems much simpler; an important aspect of our work was that the framework has been validated in real projects;
- thanks to this framework more persons can use measurement systems and customize the existing ones;
- the framework contributed to spreading the ISO/IEC 15939 standard, which was a very important change in the metric culture in the development unit.

The effect, which we identified as a result of the evaluation in this study, was that using the framework reduced the time (by ca. 50%) required to development of measurement systems and the size of measurement systems (by 35%).

One of the important aspects in our study was the use of the framework to construct a measurement system summarizing the status of all projects executed in the organization (as mentioned in Section 5.4). This measurement system is used to provide a high-level status of different projects and allows a high-level comparison of projects statuses in the development unit.

As a result of our study, we can put recommendation that in order to be effective and efficient when using metric programs we recommend adopters of ISO/IEC 15939 (which we identified as the most important factors in Section 7.2):

- Reason in terms of information needs rather than what can be measured. This usually leads to identifying stakeholder and thus the person who has a real interest in collecting metrics. This also assures that metrics are used in the decision process.
- Use indicators as the main information products provided to the stakeholders, not base or derived measures. Using indicators makes the daily monitoring efficient as the stakeholders are not required to investigate details too frequently (although the details should be provided to the stakeholders when needed).
- Use customized measurement systems rather than off-the-shelf metric tools – measurement systems should be developed to fulfill specific information needs, while metrics should be as generic as possible (so that metrics are not specific for information needs, but rather for entities being measured).
- Build own competence – train a dedicated team to work with metrics rather than involve external consultants.

However, as stressed in the paper, the first bullet is the most important one and helps to ensure that metrics are actually used and not just collected.

Although this study has been done in the context of projects executed according to RUP (Rational Unified Process) or Lean Development, we do not see those as prerequisites for successful use of this framework.

10. Conclusions

In this paper we presented a framework for building measurement systems for measuring entities in software development. The measurement systems are based on similar principles as measurement systems in other engineering disciplines, reusing such concepts as measurement instruments, base/derived measures, indicators or measured objects. The framework provides the possibility to develop measurement systems based on ISO/IEC 15939 standard in a short time, thus minimizing the effort required for setting measurement systems. As a consequence of this, quality managers and designers of measurement systems are more efficient (e.g. they can work with more metrics at the same time).

The presented framework is used in one of the development units within Ericsson, and this paper presents its positive contribution to optimizing measurement processes in the unit. The results of our evaluations in the development unit show that the framework reduces the code required for a measurement system by 35% and reduces the development time by 50% (as perceived by developers of measurement systems) compared to previous solutions in the development unit. When using this framework the development of new measurement systems does not require programming experts – a basic knowledge in metrics and company's

database infrastructure suffices. The stakeholders were given the possibility to reuse elements of measurement systems and easily share information at various levels of abstraction with each other. This, in turn, contributed to the wide-spread of measurement systems at the studied development unit within Ericsson.

The framework addresses such requirements as for example (i) ease of use, (ii) modulatiry, (iii) reuse of measurement instruments, (iv) reuse of metrics, or (v) reuse of values of metrics. These requirements were found to be important in the studied development unit.

The use of ISO/IEC 15939 standard proved itself useful in structuring metrics and measurement systems. Our recommendation is to use it when working with metrics since it requires structural thinking and identifying the most important metrics for a specific purpose.

Acknowledgements

The project has been partially sponsored by the Swedish Strategic Research Foundation (www.stratresearch.se) under the program Mobility in IT. It was also partially sponsored by Ericsson Software Architecture Quality Center. We would like to thank the involved managers at Ericsson for their support in this study. We would also like to thank the anonymous reviewers for guidance in improving our work.

Appendix A. Measurements – standard concepts

In this section we provide the definitions of the main elements in measurement systems as defined in the ISO/IEC 15939 standard, and in particular in its information model (of which an adapted version is presented in Fig. 10).

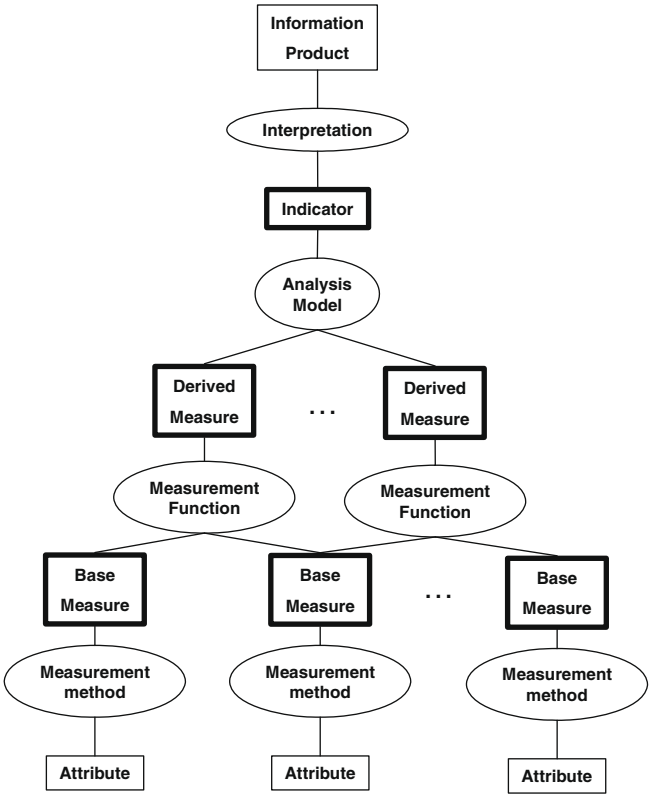

Fig. 10. Measurement information model – adapted from ISO/IEC 15939:2007.

In order to bridge the measurement theory, standardized vocabulary of metrology, and measurement in software engineering, we need to quote the definitions of the concepts important in this paper from [32]:

- *Measurement system* – set of measuring (measurement) instruments and other devices or substances assembled and adapted to the measurement of quantities or specified kinds within specified intervals of values.
- *Measuring instrument* – device or combination of devices designed for measurement of quantities. In software engineering, this term can be used to characterize most metric tools.

These concepts are used in the ISO/IEC 15939:2007 standard to define measurement processes in software engineering. This software engineering standard is a normative specification for the processes used to define, collect, and analyze quantitative data is software projects or organizations. The central role in the standard plays the *information product* which is a set of one or more indicators with their associated interpretations that address the information need. The *information need* is an insight necessary for a stakeholder to manage objectives, goals, risks, and problems observed in the measured objects (e.g. projects, organizations, software products). We use the following definitions from ISO/IEC 15939:2007:

- *Base measure* – metric defined in terms of an attribute and the method for quantifying it. This definition is based on the definition of base quantity from VIM.
- *Derived measure* – metric that is defined as a function of two or more values of base measures. This definition is based on the definition of derived quantity.
- *Indicator* – metric that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs.
- *Decision criteria* – thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result. The analysis contains a set of criteria which determine a set of notification actions which should notify the stakeholder about the change of the status of the indicator.
- *Information product* – one or more indicators and their associated interpretations that address an information need.
- *Measurement method* – logical sequence or operations, described generically, used in quantifying an attribute with respect to a specified scale.
- *Measurement function* – algorithm or calculation performed to combine two or more base measures.
- *Attribute* – property or characteristics of an entity that can be distinguished quantitatively or qualitatively by human or automated means.
- *Entity* – object that is to be characterized by measuring its attributes.
- *Measurement process* – process for establishing, planning, performing and evaluating measurement within an overall project, enterprise or organizational measurement structure.

The view on measures presented in ISO/IEC 15939 is consistent with other engineering disciplines, the standard states at many places that it is based on such standards as ISO/IEC 15288:2002 (Systems engineering – System life cycle processes) [47], ISO/IEC 14598-1:1999 (Information technology – Software product evaluation) [48], ISO/IEC 9126-x [44] or International vocabulary of basic and general terms in metrology (VIM) [32].

References

[1] N. Fenton, S.L. Pfleeger, R.L. Glass, Science and substance: a challenge to software engineers, IEEE Software 11 (1994) 86–95.
[2] S.L. Pfleeger, R. Jeffery, B. Curtis, B. Kitchenham, Status report on software measurement, IEEE Software 14 (1997) 33–34.
[3] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, International Thomson Computer Press, London, 1996.
[4] H. Zuse, A Framework of Software Measurement, Walter de Gruyter, Berlin, New York, 1998.
[5] H. Zuse, Foundations of object-oriented software measures, in: Proceedings of the 3rd International Software Metrics Symposium, IEEE, 1996, pp. 75–88.
[6] M. Umarji, H. Emurian, Acceptance issues in metrics program implementation, in: H. Emurian (Ed.), 11th IEEE International Symposium Software Metrics, 2005, pp. 10–17.
[7] A. Gopal, T. Mukhopadhyay, M.S. Krishnan, The impact of institutional forces on software metrics programs, IEEE Transactions on Software Engineering 31 (2005) 679–694.
[8] A. Gopal, M.S. Krishnan, T. Mukhopadhyay, D.R. Goldenson, Measurement programs in software development: determinants of success, IEEE Transactions on Software Engineering 28 (2002) 863–875.
[9] K.L. Atkins, B.D. Martin, J.M. Vellinga, R.A. Price, STARDUST: implementing a new manage-to-budget paradigm, Acta Astronautica 52 (2003) 87–97.
[10] J. Lawler, B. Kitchenham, Measurement modeling technology, IEEE Software 20 (2003) 68–75.
[11] T. Kilpi, Implementing a software metrics program at Nokia, IEEE Software 18 (2001) 72–77.
[12] F. Niessink, H. van Vliet, Measurement program success factors revisited, Information and Software Technology 43 (2001) 617–628.
[13] Niessink F, van Vliet H. Measurements should generate value, rather than data, in: Sixth International Software Metrics Symposium, 1999, pp. 31–38.
[14] S. De Panfilis, B. Kitchenham, N. Morfuni, Experiences introducing a measurement program, Information and Software Technology 39 (1997) 745–754.
[15] P. Goodman, Practical Implementation of Software Metrics, McGraw-Hill, London, 1993.
[16] K.H. Möller, Software Metrics: A Practitioner's Guide to Improved Product Development, Chapman & Hall, London, 1993.
[17] L. Chirinos, F. Losavio, J. Boegh, Characterizing a data model for software measurement, Journal of Systems and Software 74 (2005) 207–226.
[18] J. Schalken, H. van Vliet, Measuring where it matters: determining starting points for metrics collection, Journal of Systems and Software 81 (2008) 603–615.
[19] N.P. Kolev, S.T. Yordanova, P.M. Tzvetkov, Computerized investigation of robust measurement systems, IEEE Transactions on Instrumentation and Measurement 51 (2002) 207–210.
[20] R.v. Solingen, The Goal/Question/Metric Approach: A Practical Handguide for Quality Improvement of Software Development, McGraw-Hill, 1999.
[21] R. Jeffery, M. Berry, A framework for evaluation and prediction of metrics program success, in: First International Software Metrics Symposium, IEEE Computer Society, Los Alamitos, CA, 1993, pp. 28–39.
[22] T. Hall, N. Fenton, Implementing effective software metric programs, IEEE Software 14 (1997) 55–65.
[23] N.E. Fenton, M. Neil, Software metrics: successes, failures and new directions, Journal of Systems and Software 47 (1999) 149–157.
[24] D. Wisell, P. Stenvard, A. Hansebacke, N. Keskitalo, Considerations when designing and using virtual instruments as building blocks in flexible measurement system solutions, in: P. Stenvard (Ed.), IEEE Instrumentation and Measurement Technology Conference, 2007, pp. 1–5.
[25] G. Kai, Virtual measurement system for muzzle velocity and firing frequency, in: Eighth International Conference on Electronic Measurement and Instruments, 2007, pp. 176–179.
[26] R.F. Kunz, G.F. Kasmala, J.H. Mahaffy, C.J. Murray, On the automated assessment of nuclear reactor systems code accuracy, Nuclear Engineering and Design 211 (2002) 245–272.
[27] H. Zhiyao, W. Baoliang, L. Haiqing, An intelligent measurement system for powder flowrate measurement in pneumatic conveying system, IEEE Transactions on Instrumentation and Measurement 51 (2002) 700–703.
[28] A.N. Zaborovsky, D.O. Danilov, G. V Leonov, R.V. Mescheriakov, Software and hardware for measurements systems, in: D.O. Danilov (Ed.), The IEEE-Siberian Conference on Electron Devices and Materials, IEEE, 2002, pp. 53–57.
[29] A.S. Sehmi, N.B. Jones, S.Q. Wang, G.H. Loudon, Knowledge-based systems for neuroelectric signal processing, IEE Proceedings-Science, Measurement and Technology 141 (1994) 215–223.
[30] J. Feigin, K. Pahlavan, Measurement of characteristics of voice over IP in a wireless LAN environment, in: K. Pahlavan (Ed.), IEEE International Workshop on Mobile Multimedia Communications, 1999, pp. 236–240.
[31] M. Foote, D. Horn, Video measurement of swash zone hydrodynamics, Geomorphology 29 (1999) 59–76.
[32] International Bureau of Weights and Measures, International vocabulary of basic and general terms in metrology = Vocabulaire international des termes fondamentaux et généraux de métrologie, International Organization for Standardization, Genève, Switzerland, 1993.
[33] Predicate Logic: TychoMetrics, 2008. <http://www.predicatelogic.com> (accessed 30.06.08).
[34] F. Garcia, M. Serrano, J. Cruz-Lemus, F. Ruiz, M. Pattini, ALARACOS research group: managing software process measurement: a meta-model based approach, Information Sciences 177 (2007) 2570–2586.
[35] W.L. Oberkampf, T.G. Trucano, Verification and validation in computational fluid dynamics, Progress in Aerospace Sciences 38 (2002) 209–272.
[36] N.P. Kolev, P.M. Tzvetkov, S.T. Yordanova, Development of measurement systems with robustness feedback, in: P.M. Tzvetkov (Ed.), Instrumentation and Measurement Technology Conference, 1996, IMTC-96, Conference Proceedings, Quality Measurements: The Indispensable Bridge between Theory and Reality, vol. 2, IEEE, 1996, pp. 881–883.
[37] EventHelix: system availability and reliability. <http://www.eventhelix.com/RealtimeMantra/FaultHandling/system_reliability_availability.htm> (accessed: 10.02.08).
[38] P. Kruchten, The Rational Unified Process, Addison-Wesley, Reading, Mass, 1999.
[39] Greenlight: PROPS – project management in multi-project organization. <http://www.greenlightpm.com/props.htm> (accessed 15.02.08).
[40] C. Robson, Real World Research, Blackwell Publishing, Oxford, 2002.
[41] M. Staron, W. Meding, Short-term Defect Inflow Prediction in Large Software Project – An Initial Evaluation, International Conference on Empirical Assessment in Software Engineering (EASE), British Computer Society, Keele, UK, 2007.
[42] M. Staron, W. Meding, Predicting Monthly Defect Inflow in Large Software Projects – An Industrial Case Study, International Symposium on Software Reliability – Industry Track, IEEE, Trolhattan, Sweden, 2007.
[43] M. Staron, W. Meding, Predicting weekly defect inflow in large software projects based on project planning and test status, Information and Software Technology (2007).
[44] International Standard Organization, International Electrotechnical Commission: ISO/IEC 9126 – Software engineering – Product quality Part: 1 Quality model. International Standard Organization/International Electrotechnical Commission, Geneva, 2001.
[45] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslèn, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publisher, Boston MA, 2000.
[46] W.M.K. Trochim, Research methods knowledge base, 2008. <http://www.socialresearchmethods.net/kb/> (accessed 30.06.08).
[47] International Standard Organization, Systems engineering – System life cycle processes 15288:2002, 2002.
[48] International Standard Organization, Information technology – Software product evaluation 14598-1:1999, 1999.