Lessons Learned on Systematic Metric System Development at a large IT Service Provider

Matthias Vianden*, Horst Lichter†
Research Group Software Construction
RWTH Aachen University
{matthias.vianden*, horst.lichter†}@swc.rwth-aachen.de

***Abstract*—Even though a lot of work was contributed to extend and enhance metric requirements gathering techniques, metric systems are often developed chaotically and a solid dedicated metric system engineering approach is still missing. This paper provides our experiences at developing a metric system together with a large IT service provider and presents an overview on our reference architecture for enterprise measurement infrastructures. Furthermore we give some insights into our metric systems engineering approach which integrates software engineering best practices, modern ideas like micro services, and well established metric related techniques such as GQM.**

***Index Terms*—Metric Systems; Reference Architecture; Field Study**

I. Introduction

By analyzing metrics, process managers are able to identify processes that contribute to project success or failure. Therefore, process improvement models such as CMMI encourage software development organizations to build up abilities to systematically apply metrics and measure the quality of the development processes and software systems [1]. This makes metrics the key necessity for objective process and product optimization. However, the research community agrees that it is often difficult to find the ”right” metrics and provide ”good” measurements.

Well established methods like Basili's GQM [2] or its modern variations like GAM [3] may help to gather requirements for metrics and dashboards but they are far away from a complete engineering approach which should include requirements engineering, development, testing, operation, and maintenance. Münch and Heidrich successfully worked on metric dashboards [4] and proposed a GQM based development method [5]. Unfortunately, this and several other proposed approaches follow a waterfall processes model and tend to not utilize modern software engineering ideas such as incremental and iterative development, SOA, or prototyping. Also the maintenance and operation phase is not addressed.

We have organized this paper as follows. Section 2 presents the basic concepts of metric systems. We present the core ideas of a reference architecture for enterprise measurement infrastructures in section 3 followed by a description of the environment for the presented field study in section 4. Section 5 gives an overview of the main stakeholders that our metric system engineering approach addresses. We describe the most important activities and tasks of the development process that we applied during the field study in section 6. Section 7 lists major recommendations extracted from this field study. Finally, section 8 concludes this paper.

II. Core Concepts

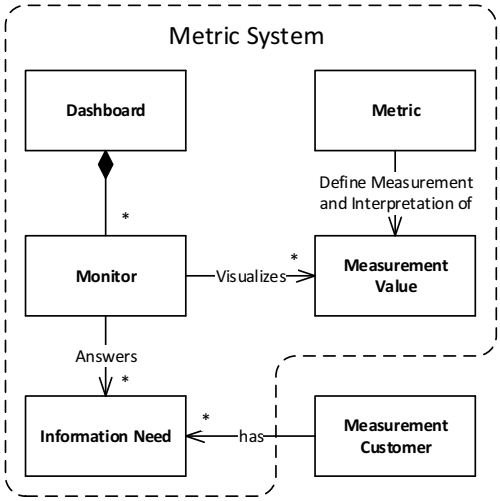

Fig. 1. Relationships of the central metric system concepts

Fig.1 depicts the central metric system concepts and relationships. It is a simplified version of a common metric and measurement ontology, e.g. the ones published by Olsina, Staron, and many others [6]–[9], and aligns well with ISO 15939 [10]. The core of a metric system are *dashboards* containing *monitors* that answer *information needs* of a *measurement customer*. Monitors visualize *measurement values* from various sources. The measurement or calculation of the values as well as their interpretation is defined by a *metric*.

Unlike GQM we don't believe that the information needs should be fairly distinct from each other. On the one hand, two different monitors can answer the same need. For example the information need ”How is the ratio between known and addressed defects?” could be answered by a monitor showing a bar chart of the new, known, and answered defects per week and a monitor showing a bullet graph of the open defects. On the other hand, an information need may be only answered by a combination of monitors. E.g., the information need ”If we are behind the planned schedule, are staffing problems the reason?” could be answered by a monitor showing a bullet

Software Engineering Conference (APSEC), 2014 21st Asia-Pacific , vol.2, no., pp.47,50, 1-4 Dec. 2014
doi: 10.1109/APSEC.2014.92

graph for the current schedule variance and a second monitor showing a list of staffing problems per day.

III. EMI Reference Architecture

From the core concepts in the previous section we derived a flexible, specific, and micro service-based reference architecture for enterprise measurement infrastructures (EMIs) [11], [12]. The most important services and layers are depicted in Fig.2.

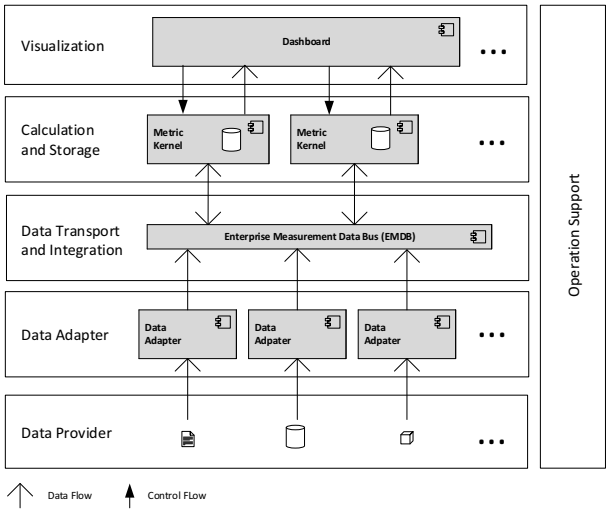

Fig. 2. Condensed static view on the EMI reference architecture

The information needs are addressed by specialized analysis or dashboard tools in the *Visualization Layer*. The data required for the calculation of metrics is provided by systems in the *Data Provider Layer* in the bottom. These systems are connected to the infrastructure using *Data Adapters* in the *Data Adapter Layer*.

Visualization tools often require aggregated information besides pure base values. This information is produced and provided by services in the *Calculation and Storage Layer*, so called *Metric Kernels*. The *Data Transport Layer* realizes a common communication infrastructure for all services of the Calculation and Storage Layer and of the Data Provider Layer. The *Operations Support Layer* contains services to operate and monitor the infrastructure.

IV. Environment

This field study was conducted in co-operation with a large IT provider for insurance companies which needs to deal with legacy systems as well as provide modern services. As this IT provider is CMMI Level 3 certified, all development projects need to apply the organization's standard development process. We worked together with the engineering-process group and were supported by two metric experts from within the company. In the later stages of the field study the company started a dedicated project to support the development and pre-production stages of the metric infrastructure including development and operation teams as well.

V. Stakeholders

A metric system needs to address the requirements of different stakeholders. Based on a literature review and our former experience we identified the following five main stakeholders each providing a specific set of requirements regarding the architecture and the resulting measurement infrastructure.

Measurement Customer: A typical example of a measurement customer is a project manager who is interested in the actual status of a project. Measurement customers have a brought variety of information needs. Unfortunately, the answers to the different information needs are often stored in different tools. Typically, the information needs of measurement customers change over time. Especially reorganizations lead to new and changed responsibilities of measurement customers which inevitably lead to changes in information needs. Hence, a metric system and its development process has to support the evolution (i.e. constant change) of metrics, integrated systems, and visualizations.

Metric Expert: The metric expert is responsible to assist metric customers in finding the right metrics, maintaining metric best practices and managing organizational wide metrics and measurement programs. Typically, large organizations have dedicated metric experts whereas in smaller ones process managers, software architects, or lead developers fill up this role. Metric experts like to provide common and generic solutions to the metric customers that best fit their needs.

Architect: They are responsible to design the metric services and the actual enterprise measurement infrastructure using established reference architectures and concepts as guides. They require a broad set of tools and concepts to deal with the integration of different systems and data and also like to maintain and use clear guidelines for architectural decisions.

Developer: They implement metrics, visualizations, and tools to gather data. They need a clear structure of all specific tasks during the metric system development. Furthermore they need development support for debugging during development as well as core services like a dedicated logging service. The micro service-based reference architecture requires that the development process addresses different integration stages in order to allow the developer to test the metric services incrementally.

Operator: This stakeholder is often ignored when designing and implementing metric systems. The operations department's most important task is to guarantee that all (metric) systems are working inside their operational parameters. This requires a set of operation tools as part of a metric system as well as addressing these operation needs in the requirements phase for the metric system.

VI. Development Process

Obviously, using a specific reference architecture implies specific steps in the development process. These steps align with the services in the layers of the reference architecture, most importantly the dashboard and visualization tools that satisfy the information needs. As it is always hard to define the right metrics and visualizations, we performed several

Software Engineering Conference (APSEC), 2014 21st Asia-Pacific , vol.2, no., pp.47,50, 1-4 Dec. 2014
doi: 10.1109/APSEC.2014.92

iterations on prototypes. This provided a solid base for the construction of the metric system. The construction itself was cut into increments to decrease complexity and receive early feedback. As mentioned before we involved the operation department early and addressed their requirements by specialized operation services in the metric infrastructure.

A. Requirements Analysis and Conception

Fig.3 depicts the main steps of the requirements phase and the two construction increments of the metric system. The three main requirements sub-processes are located in the left. They were executed iteratively (indicated by the loop-icon). Additionally, the whole requirements process could be executed iteratively as well if excessive flaws would be detected during the "Prototype and Evaluate" or the "Plan Increments" sub-process.

The requirements gathering sub-processes started with an assessment of existing metric systems used by the IT service provider. Because we noticed that the information needs of the project managers were changing we conducted interviews to systematically gather these changes [13]. We then analyzed the changed information needs and developed a prototype for a new metric-based monitoring dashboard which was evaluated by the project managers. Furthermore, we developed some prototypes focusing on specific monitors. After some iterations we arrived with the central visualizations and diagrams required for the project managers. Then we analyzed all gathered requirements and specified the realization increments. The first one focused on metrics to analyze project risks, the second focused on metrics based on error and enhancement tickets.

B. Risk Metrics Increment

We started the increment by identifying and planning the design of the required metric services (data adapters, metric kernels, visualizations, and dashboards). The design was again performed iteratively. We first created a rough version of the EMI and specifically focused on the integration part. Then, we started the detailed design of the required data adapters and metric kernel. In parallel we conducted several workshops to discuss possible failures and exception behavior in order to define meaningful test scenarios and test cases. We evaluated the design and did some minor changes after the first iteration.

Then we started the construction of the metric services. The integration of the developed services worked flawless due to the good and thorough design before we started construction. Every service could be tested in a local EMI environment. We also tried to continuously deploy the current versions of the metric services to a pre-production environment. This enabled continuous testing by the metric experts and provided important feedback to the developers. This construction of the metric services was finished after 1.5 months and we were able to release a first version and start pre-production tests with metric customers.

C. Ticket Metrics Increment

The focus of the second increment was to implement a dashboard for monitoring ticket-based metrics. It was also started by identifying the required metric services. Unfortunately, due to resource constraints the design activities were skipped. Hence, the developer had to perform the design "on the fly". The local development of the metric services again worked smoothly due to the well designed reference architecture and development and operation tool support. The deployment and test in the pre-production environment, however, did not work as flawlessly as anticipated. The reasons were problems with the deployment configuration and incompatible interfaces as well as configuration issues within the dashboard services. The development also took longer than anticipated. A lot of these problems, from our point of view, are due to the missing design phase.

VII. Recommendations

Throughout the field study we gathered a lot of experience with the development of metric systems. In this section we present the most important lessons learned supporting an effective an efficient development of metric systems.

Apply appropriate requirements gathering techniques! Mind mapping, for example, is a appropriate technique to order and arrange goals and information needs. This is especially helpful in workshops. We also recommend using mind maps to document information needs because they allow space saving documentation of hierarchical data.

Involve metric experts from the beginning! We recommend that at least one metric expert participates in the interviews or workshops because metric experts become particularly useful if the discussion is to one-sided or stuck.

Apply best practices to identify metrics! We typically use GQM to analyze the information needs. However, we very rarely use the formalized goal definition as this typically leads to unnecessary and narrow discussions. Furthermore, we try to align the metrics with the measurement information model of ISO 15939. We recommend to keep your set of metrics as simple and small as possible without sacrificing metrics for a dedicated information need.

Develop prototypes iteratively and incrementally! We recommend developing the monitors incrementally one after the other. Each monitor itself should be developed iteratively. After a new monitor is added the dashboard has to be evaluated to ensure that all monitors together cover the information needs addressed so far.

Always perform a design phase! This is very obvious and well known! Especially if the development team does not have a solid understanding of the the reference architecture it is important to include a design phase and to evaluate the designs.

Provide tool support and frameworks! Using a framework according to a reference architecture can drastically reduce the development effort and increase reference architecture compliance. It supports the development with

Software Engineering Conference (APSEC), 2014 21st Asia-Pacific , vol.2, no., pp.47,50, 1-4 Dec. 2014
doi: 10.1109/APSEC.2014.92

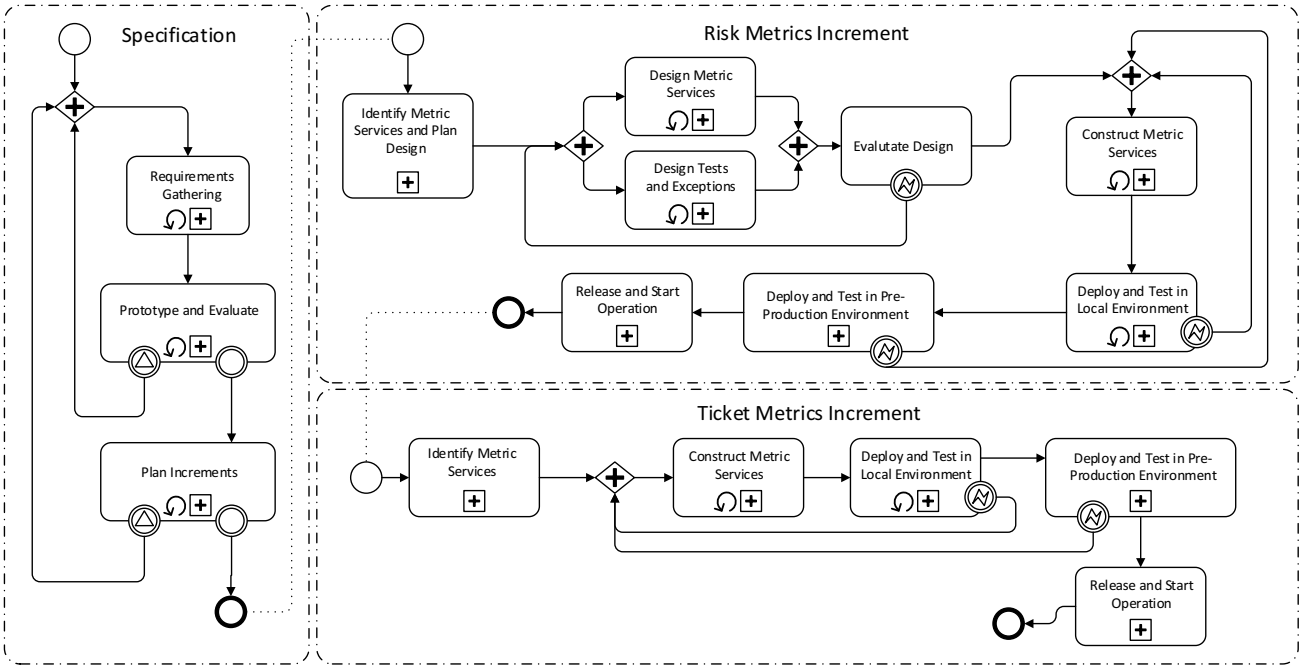

Fig. 3. BPMN diagram of the main process steps used to develop the metric system at the IT service provider

dedicated hot-spots and pre-fabricated solutions for typical problems and can provide ready-to-use services as well.

VIII. Conclusion

We received a lot of positive feedback from the metric customers on the thorough requirements gathering phase and inclusion of several dashboard and monitor prototypes. The resulting solid requirements also eased increment planning and design as well as construction of the two increments. On the one hand, we described the problems which raised in the second increment when skipping the design phase. On the other hand, we also saw that a solid reference architecture and supporting framework as well as ready-to-use services ease and streamline the development even without a solid design as a foundation.

We are currently conducting further field studies utilizing our metric systems engineering approach (i.e. reference architecture and process model) to further improve our approach. We are also enhancing our framework to a full-fledged software development kit (SDK) for multiple development platforms to support the construction of enterprise measurement infrastructures even further.

References

[1] C. P. Team, "CMMI for Development, Version 1.3 CMMI-DEV, V1.3," Tech. Rep. November, 2010.

[2] V. R. Basili, "Software modeling and measurement: the Goal/Question/Metric paradigm," p. 24, 1992. [Online]. Available: http://portal.acm.org/citation.cfm?id=137076

[3] L. Cyra and J. Górski, "Extending GQM by Arguement Structures," in *CEE-SET 2007*, vol. 44, no. 5. Springer, Sep. 2008, pp. 26–39.

[4] J. Heidrich and J. Münch, "Software project control centers: concepts and approaches," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 3–19, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0N-49KH2NM-4/2/b4f711c4ad0a0614f67ea2a1b1947eee

[5] ——, "Goal-oriented setup and usage of custom-tailored software cockpits," in *PROFES '08: Proceedings of the 9th international conference on Product-Focused Software Process Improvement*. Springer-Verlag, 2008, pp. 4–18. [Online]. Available: http://www.springerlink.com/index/e13058u041024759.pdf

[6] L. Olsina and M. D. L. A. Martín, "Ontology for Software Metrics and Indicators: Building Process and Decisions Taken," in *Web Engineering*, 2004, p. 778. [Online]. Available: http://www.springerlink.com/content/vk22y143djyhpgrx

[7] M. Staron and W. Meding, "A Modeling Language for Specifying and Visualizing Measurement Systems for Software Metrics," 2007, pp. 300–307.

[8] J. A. McQuillan and J. F. Power, "Towards re-usable metric definitions at the meta-level," in *PhD Workshop of the 20th European Conference on Object-Oriented Programming (ECOOP 2006)*, 2006.

[9] L. Chirinos, F. Losavio, and J. Bø egh, "Characterizing a data model for software measurement," *Journal of Systems and Software*, vol. 74, no. 2, pp. 207–226, 2005.

[10] ISO, "ISO/IEC 15939: 2002 Software Engineering - Software Measurement Process," International Organization for Standardization, Geneva, Switzerland, Tech. Rep., 2002.

[11] M. Vianden, H. Lichter, and A. Steffens, "Towards a Maintainable Federalist Enterprise Measurement Infrastructure," in *Joint Conference of the 23nd International Workshop on Software Measurement (IWSM) and the 8th International Conference on Software Process and Product Measurement (Mensura)*, Ankara, Turkey, 2013.

[12] M. Vianden, "EMI Homepage," 2014. [Online]. Available: http://www.enterprise-measurement.com

[13] M. Vianden, H. Lichter, and S. Jeners, "History and Lessons Learnt from a Metrics Program at a CMMI Level 3 Company," in *Proceedings of 20th Asia-Pacific Software Engineering Conference, APSEC 2013, Vol. 2*, no. CMMI, 2013.

Software Engineering Conference (APSEC), 2014 21st Asia-Pacific , vol.2, no., pp.47,50, 1-4 Dec. 2014
doi: 10.1109/APSEC.2014.92