2014 21st Asia-Pacific Software Engineering Conference

Experience on a Microservice-based Reference Architecture for Measurement Systems

Matthias Vianden, Horst Lichter, Andreas Steffens
Research Group Software Construction
RWTH Aachen University
Aachen, Germany
{Matthias.Vianden, Horst.Lichter, Andreas.Steffens}@swc.rwth-aachen.de

***Abstract* — In our former work we proposed a microservice-based reference architecture for Enterprise Measurement Infrastructures (EMI) which received encouraging feedback. The reference architecture supports the systematic development of measurement systems. This paper provides deeper insight into the application of the reference architecture by presenting the results of two field studies after an examination of the most important requirements that drove the development of the reference architecture. The two selected field studies were conducted with large cooperation partners from industry and research and addressed real problems. Using our reference architecture, development process, and requirements gathering technics we were able to successfully build the EMIs presented in this paper. These results further ease the application of microservice inside our reference architecture and support practitioners with specific examples.**

Keywords — Reference Architecture; Microservice; SOA; Field Study; Measurement System;

I. INTRODUCTION

Improvement reference models such as CMMI require software development organizations to build up abilities to systematically apply metrics to support project management [1]. Based on quantifiable metrics process managers are able to identify processes that contribute to project success or failure. Hence, metrics are a necessity for objective process optimization. However, it is often difficult to integrate measurement values from a large variety of different software systems used in software development projects.

Resulting in the different application scenarios for dashboards and measurement systems (strategic, analytical, or operational [2]) modern measurement systems use new integration approaches. Most recently, considerable research was devoted to using service oriented (SOA) and agent based architectures for measurement systems [3]. New loosely coupled integration architectures are researched in the area of enterprise architecture integration (EAI) ([4]–[9]).

Unfortunately, these ideas are not systematically used to build flexible, maintainable, and robust measurement infrastructures. Most of the solutions found in the industry right now are based on BI (Business Intelligence) systems. However, all of the proposed solutions (even the new SOA and agent based approaches) use a central repository to store and integrate the measurement data. Hence, they suffer from well-known centralized integration problems; like the need for a common data schema and the well-known schema mapping problem. All of these leads to stiff, unmaintainable, and fragile measurement systems which are very rarely altered. Additionally, not every data should or can be measured (and stored) by means of relational data schemata [10]. Our reference architecture for Enterprise Measurement Infrastructures (EMIs) [11] tackles these core problems using dedicated microservices for measurement, calculation and visualization.

We organized this paper as follows. Section II reflects on the requirements for the reference architecture and describes the central roles involved with enterprise measurement infrastructures. Section III describes the core design or the reference architecture. Section IV.a and IV.b contain two selected field studies on the application of the reference architecture. We summarize and conclude the paper in section V.

II. REFERENCE ARCHITECTURE REQUIREMENTS

A measurement infrastructure needs to address the requirements of different stakeholders. After an elaborated examination of the literature and based on our experience we identified five main stakeholders. Each of these stakeholders provides a unique and specific set of requirements regarding the architecture and resulting measurement infrastructure.

A. Measurement Customer

A project manager is a typical example of a measurement customer. She is interested in the actual status of her project and does not care (and should not!) about the technology behind data collection, integration, calculation, and visualization. Measurement customers have a brought variety of information needs. Unfortunately, the answers to the different information needs are typically stored in many different tools.

Most importantly, measurement customers demand correct and up-to-date data because old or incorrect data leads to wrong conclusions and wrong decisions. Hence, an EMI should provide mechanisms that guarantee a fast recognition and processing of relevant events. Additionally, up-to-date data requires robustness and high availability of the EMI.

Our experience with many industry partners shows that the information needs of measurement customers often change over time. For example, development tools are replaced by other tools or processes and organization schemas are changed. Especially reorganizations lead to new and changed responsibilities of individual measurement customers and

1530-1362/14 $31.00 © 2014 IEEE
DOI 10.1109/APSEC.2014.37

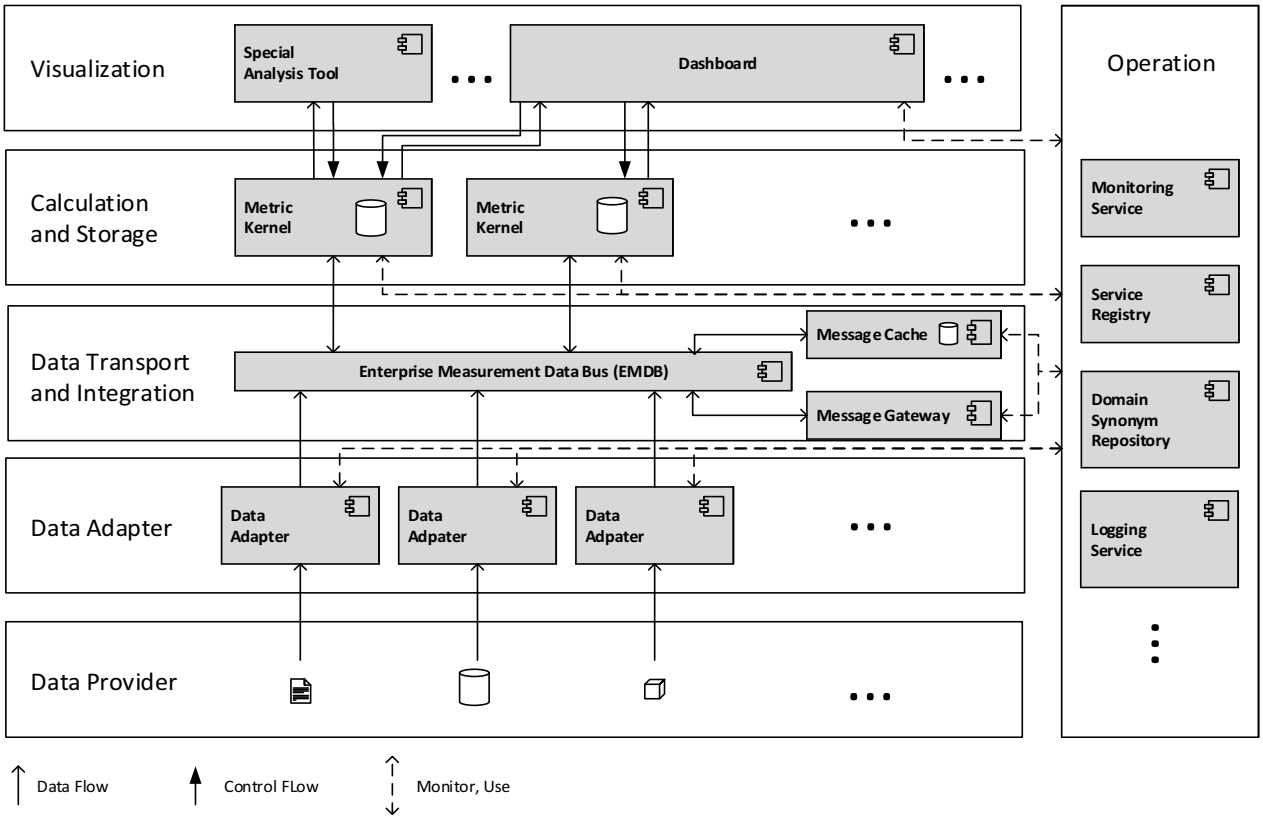

Fig.1. Enterprise Measurement Infrastructure (EMI) reference architecture services, layers, data flow, control relations, and monitor/use relations

roles which inevitably lead to changes in information needs. Concluding from this, another important requirement for an EMI is to support the evolution of metrics, integrated systems, and visualizations.

B. Metric Expert

The responsibility of the metric expert is 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 organizations 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. This often requires to integrate data from a lot of different tools. They are also responsible for specifying requirements for the metric applications.

C. Architect

Architects are responsible for the design of the metric services and the actual enterprise measurement infrastructure. They use established reference architectures and concepts as guides during the specification phase. They require a broad set of tools and concepts to deal with the integration of different systems and data. They also like to maintain and use clear guidelines for architectural decisions. During the design of a metric services they need to work closely with metric experts who specify the requirements for the services. They also need to include additional requirements regarding operation from the operators.

The architect needs to be able to design an EMI that integrates different tools in a way that a comprehensive and flexible calculation of metrics is possible. To achieve this goal, the infrastructure has to cope with the heterogeneity of those tools. Heterogeneity appears on various levels. Wache et al. [12] define structural and semantic heterogeneity of data. Structural differences lead to the problem of schema-mapping, a quite well known field of research in the database community [13]. Hence, data heterogeneity is challenging for a successful integration of those systems as well. The architects are also responsible for the alignment of an EMI to the system landscape of the organization. Hence, the infrastructure should be compatible with service oriented architectures found in modern organizations.

D. Developer

The developer implements metrics, visualizations and tools to gather data. The reference architecture needs to support the developer with a clear structure and concepts for all specific tasks. A developer also requires extensive development support for debugging during development. One of the core services required is a logging service to quickly access log information from all services. The task to integrate a new system into the infrastructure to gather its data is completely different from the implementation of a new metric calculation algorithm or the implementation of a new visualization.

Hence, a requirement for an EMI is the clear separation of system integration, calculation, and visualization.

E. Operator

This role is often ignored while building and conceptualizing or measurement infrastructures. The operations department has two main responsibilities. First, it has to guarantee that all systems are working inside their operational parameters. This requires a dedicated set of operation tools as part of an EMI. The infrastructure should at least provide or support a monitoring tool which allows analyzing the amount of data that is transported and stored in the infrastructures components. Second, the operations department has to solve upcoming problems in the infrastructure without disturbing the integrated systems as these systems are often of crucial importance for the company.

F. Requirements List for the Reference Architecture

The sections above motivate the following main functional and non-functional requirements for an enterprise measurement infrastructure:

- **R1.** Integration of heterogeneous systems to provide the basis for different metrics and visualizations.
- **R2.** Fast and up-to-date recognition and update of the metrics on a change in an integrated system.
- **R3.** Clear separation of system integration, calculation and visualization.
- **R4.** Be robust to avoid a complete system failure if a small part of the system fails. Additionally, the failure of the infrastructure should not result in a failure of the integrated systems.
- **R5.** Be flexible to support evolution of metrics, integrated systems, and visualizations.
- **R6.** Offer dedicated operation and development support.

III. Enterprise Measurement Infrastructure (EMI) - Architecture and Components

The previous set of requirements will inevitably lead to a loosely coupled federalist (decentralized and distributed) infrastructure. Especially due to requirements R4 and R5 we decided not to rely on a central database to store all measurement values and no central data schema to avoid schema-mapping problems. Each service should decide for itself what to store. This leads to some data redundancy in the infrastructure which is fine because this makes the infrastructure more robust. Also we decided to favor small micro services [14] over large full scale SOA services because they drastically improve maintainability, robustness, and flexibility of the infrastructure and seem to be very successful during the last years [15], [16].

The core services and layers of the reference architecture for enterprise measurement infrastructures (EMIs) is depicted in figure 1. The information needs of different measurement customers are addressed by specialized analysis or dashboard tools in the *Visualization Layer* at the top. The actual data needed to calculate metrics is provided by different systems in the *Data Provider Layer* in the bottom. These systems are connected to the infrastructure using dedicated *data adapters* (R1).

Visualization tools often require complex and aggregated information besides pure base values. This information is produced and provided by specialized services in the *Calculation and Storage Layer*, the 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 Layer* contains services that ease operating and monitoring the infrastructure (R6). In the following we recapitulate the core concepts of the EMI reference architecture.

A. Dataflow and Core Services

The EMI data flow depicted in Fig. 2 is based on the ISO 15939 standard. In this ISO standard data always flows from base measures to derived measures which are then combined in an analysis model to form an indicator that answers a particular information need. In the EMI we added important extensions, since even base measures (provided by data adapters) can form indicators (e.g. data from a tool like Sonar). Most importantly, derived measures can not only use base measures but the results of other derived measures as well as a combination of the two.

The Enterprise Measurement Data Bus (EMDB) realizes an event-based integration between the services of an EMI [17], [18]. Most importantly it transports measurement messages between data adapter and metric kernels. To support separation of concerns, an EMDB can implement several publish/subscribe channels for different integration tasks. The *Message Cache* is a central infrastructure service strongly coupled to the EMDB. It stores all measurement messages for operation tasks like the setup of a new Metric Kernel or data recovery (R5, R6). The Message Cache together with the Message Gateway also provide the backbone for systematic system and integration testing of the Metric Kernels and Data Adapters.

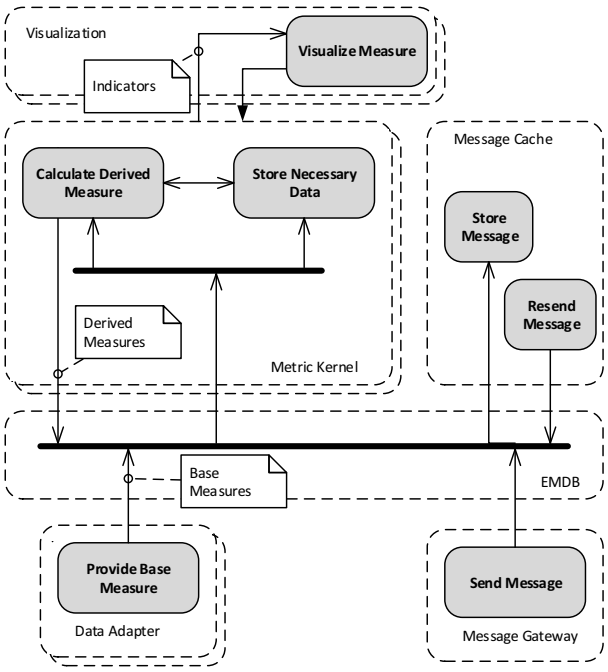

Fig. 2. Measurement and data flow in the EMI reference architecture

B. Data Adapter Patterns

The heterogeneity of the systems that are integrated into the infrastructure calls for flexible data provision mechanisms (R1, R3, and R5). Additionally the data from the data providers should be integrated as fast as possible (R2). We developed the following three core data adapter pattern to work as design guides for concrete data adapters.

- **Push-Forward**
  The data is pushed to the EMI using a Gateway as data adapter and a plugin in the data provider. The Gateway typically just sends the received data as a specific message over the EMDB. However, it can also perform additional consistency checks. The gateway typically exposes a REST API [19] to allow easy and uniform access from different data provider. This adaption pattern has the lowest latency between data change in the data provider and data arrival in the EMDB (R2).

- **Pull-Forward**
  The data is pulled from the data provider to the EMI in defined intervals (e.g. every five minutes). This adaption mechanism is used if the data provider does not support a plugin mechanism that is called whenever some data is changed. However, it obviously requires an API to access the data from the data provider. Ideally, the API allows to access only the changed data during the last interval. Due to typical intervals in the range of some minutes this adaption pattern has the highest latency.

- **Invoke-Push**
  This is to a certain extend a combination of the two patterns from above. The data is pulled from the data provider like in the pull-forward pattern. The trigger for the pull, however, is not a timer but another message or event on the EMDB that got adapted (optimally pushed) before. The latency depends on the latency of the triggering message. If it is pushed the latency is also very low.

More details on the EMI reference architecture and additional details and documentation can be found on the EMI webpage and former publications [11], [20], [21].

IV. Application of the EMI Reference Archictecture in selected Field Studies

In the following sections we present the results of two field studies that we conducted with different industrial and academic cooperation partners using the EMI reference architecture. Each section starts with a short introduction of the environment and the main requirements for the specific EMI. Afterwards we provide some details on the respective EMI design. We conclude each field study with a condensed list of gained experiences.

A. Project management metrics for SSELab

SSELab is a management infrastructure mainly supporting software development projects at RWTH Aachen University [22]. It integrates key services like version control systems (git and SVN), wikis, and change request management systems (TRAC) into one coherent platform. Currently SSELab hosts over 700 projects. Our analysis showed a very heterogeneous project environment which is dominated by software development projects. However, SSELab is also used for the administration of organizational projects and scientific projects such as paper or theses writing projects as well as teaching projects like lab courses. Even though SSELab offers a lot of features and functionality it lacks the support for measurements. Hence, the goal of this field study was to investigate what metrics are required for the project managers of SSELab projects and to develop a maintainable, robust, and flexible measurement infrastructure that enables the calculation and measurement of the metrics.

1) Development Process

The main goal was to develop a metric-based monitoring dashboard template for project managers. We conducted interviews as requirements gathering technique to get a broad feedback on the information needs of different project managers. We managed to interview 11 project managers. We used a questionnaire to keep the managers focused on the important aspects. However, some of the most interesting information needs where not directly related to our questions.

Afterwards, the results of the interviews were integrated into a large mind map following the GQM principle. From this we derived key information needs and their corresponding metric-based monitors to be included in the dashboard prototype depicted in figure 5. This prototype is designed following the dashboard design principles proposed by Few [23], [24] with the most important visualizations (Bullet Graphs) on the top left. The dashboard features visualization of source code metrics and statistical metrics on version control activities (e.g. number of commits per week) and issue tracking (e.g. number of open and closed issues per week). We then sent the dashboard prototype to the project managers to ask for feedback and conducted follow-up interviews with selected project managers.

Fig. 5. SSE-Lab metric-based monitoring dashboard prototype

Based on the feedback we changed the user interactions and updated some of the metrics and visualizations. The modified prototypes were again evaluated by the project managers. We then started the design phase because everybody was quiet satisfied with the new design. The goal of this phase was to come up with an EMI-based design for the integration of the data providers and calculation of the metrics [25].

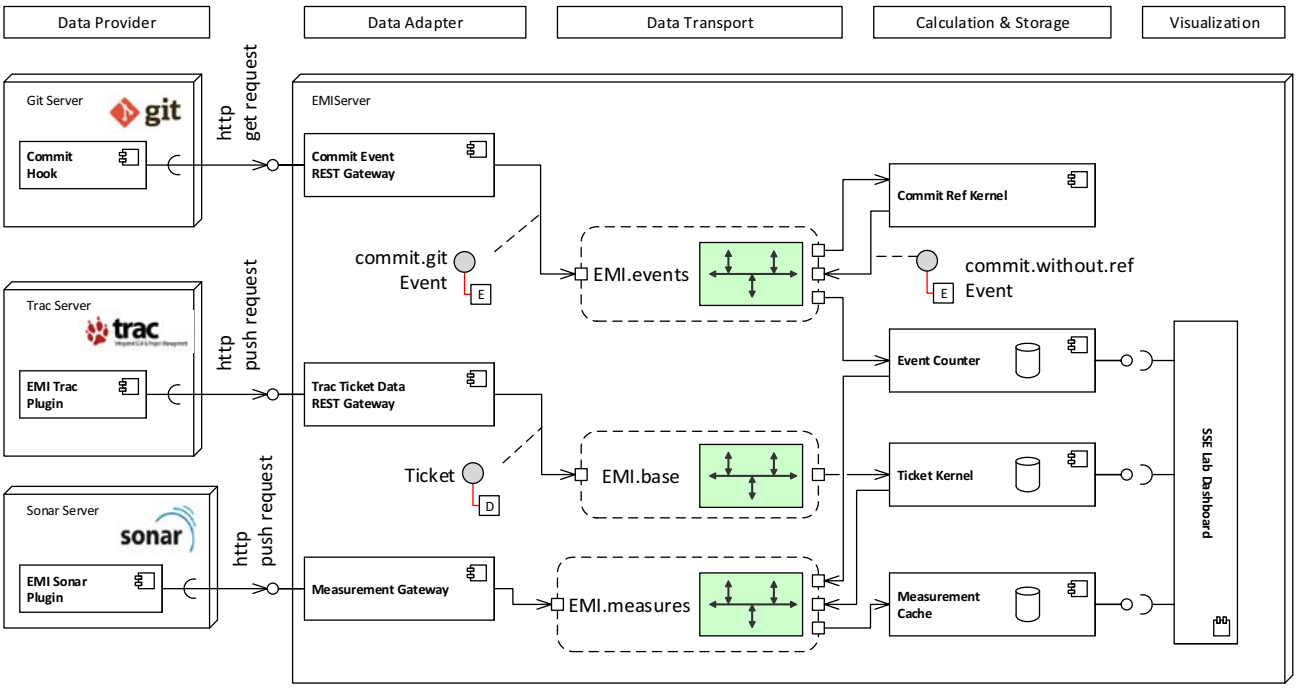

Fig. 6. Static architecture overview of the EMI for SSE-Lab

Based on these results we iteratively and incrementally developed the data adapters, the integration layer (messages) and the metric kernels. We started with the integration and data adaption on the version control systems and then moved to the issue tracker (TRAC). We also implemented a special dashboard application which can be integrated seamlessly into the SSE Lab architecture. After each increment we did a quick evaluation of the metrics and dashboard in a test environment.

2) SSE Lab Enterprise Measurement Infrastructure

The EMI for SSE Lab is depicted in figure 6. Contrasting the layout of the reference architecture in figure 1 the data flow in this figure is turned 90° going from left to right.

The three central data providers: git version control system, TRAC issue tracking, and sonar qube for source code metrics are located on the left. The data adapters for these systems implement the push-forward adapter pattern (see III.b). We build specific EMI-plugins for TRAC and sonar which hook into existing extension points in the two systems to call the respective REST-API in the data adapter on data change in the data provider. For git we used a bash script that is called in a commit hook to call the REST-API of the commit gateway.

The SSELab EMI utilizes three busses from the EMDB. The event bus (EMI.events) to transport event data, the base bus (EMI.base) to transport almost raw data (base measures) from the data providers, and the measures bus (EMI.measures) to transport measurements (derived measures).

The commit event gateway on the top left emits commit event messages to the event bus of the EMI. These messages are transported to two metric kernels: Commit Reference Kernel and Event Counter. The latter is a generic component that calculates a number of count metrics (number of event Y per X) on events on the event bus.

The Commit Reference Kernel simply checks the commit message for references to issues and does not store any data (just static checking, no semantic check!) and then again utilizes the event counter with a different event to count the commit messages without references. This shows the flexibility and reuse potential of the EMI microservices.

The TRAC ticket data rest gateway produces ticket messages on the base bus of the EMI. These tickets are analyzed by the TRAC Kernel. This kernel stores the tickets and implements the count metrics (e.g. open tickets per week) as well as the statistical analysis that feeds the box plots. The metric calculation results as well as the calculations from sonar are feed to the measures bus of the EMI and stored in the measurement cache.

As mentioned before, we built a special metric-based monitoring dashboard to visualize the metrics in SSELab. The dashboard accesses the metrics from the metric kernels using their specific REST-APIs. This allows to include some configuration for the monitors in the dashboard. For example the calculation algorithm for ticket statistics can be selected. The dashboard then simply feeds the calculation results to the visualizations.

The pre-production version of this EMI was operated on a JavaEE server in our test environment which also hosted the local databases for the metric kernels. This server was operated in a secure environment to ensure data privacy. Thanks to the EMI Monitoring Service it was always very easy to check the current status of the EMI.

3) Experiences

We gained a lot of experience with using the reference architecture on this project. The core expiries where:

- **Finding the *right* metrics is hard**
  Even though we did extensive prototyping and conducted several interviews with the metric customers (SSELab project managers) we found some problems with the metric specifications when we started implementing the metric kernels. Particularly the statistical ticket analysis in the Ticket Metric Kernel was not well defined and we were struggling with what alternative to use. In the end the flexibility of the reference architecture allowed us to implemented all the different options in the metric kernel side-by-side and allow the user to select the calculation mechanism via a specific REST-API on the Ticket Metric Kernel.
- **Microservices in an EMI are very easy to reuse**
  We developed the Event Counter and the Commit Event Gateway before we started the development on the SSELab EMI. During the specification of the EMI for SSELab we realized that we can reuse these services and it worked fluently without any major issue! It was also very easy to add additional functionality (counting commits without references) by simply adding another service.
- **An EMI is easy to maintain and to operate. We also did not have any major performance problem**
  During the iterative and incremental development of the EMI for SSELab it was very easy to extend and maintain the services. Thanks to the EMI monitoring service it was always very easy to check the status of each service and to investigate performance of the services and the EMI. We never experienced any problem with performance whatsoever. However, the production use of EMI in SSELab is still ahead of us.

B. *Project risk metrics at a large IT provider*

This field study was conduction together with a large full service IT provider for insurance companies which needs to deal with the legacy of very old systems as well as provide modern services. While some development projects can be quite small (about 100 person days), others are very large (up to 35.000 person days). As the IT provider is CMMI Level 3 certified, all development projects need to stick to the organization's standard development process which is a tailored Rational Unified Process.

This process contains a template for a metric-based monitoring dashboards designed for project managers. Because we noticed that the information needs of the project managers were changing we initiated interviews with five of them to systematically gather these changes [26].

1) Development Process

First, we analyzed the changed information needs and developed a prototype for a new metric-based monitoring dashboard. This prototype was then evaluated by the project managers that participated in the initial interviews. Then we analyzed all identified requirements and specified the increments for the realization phases. The first increment focused on metrics to analyze project risks.

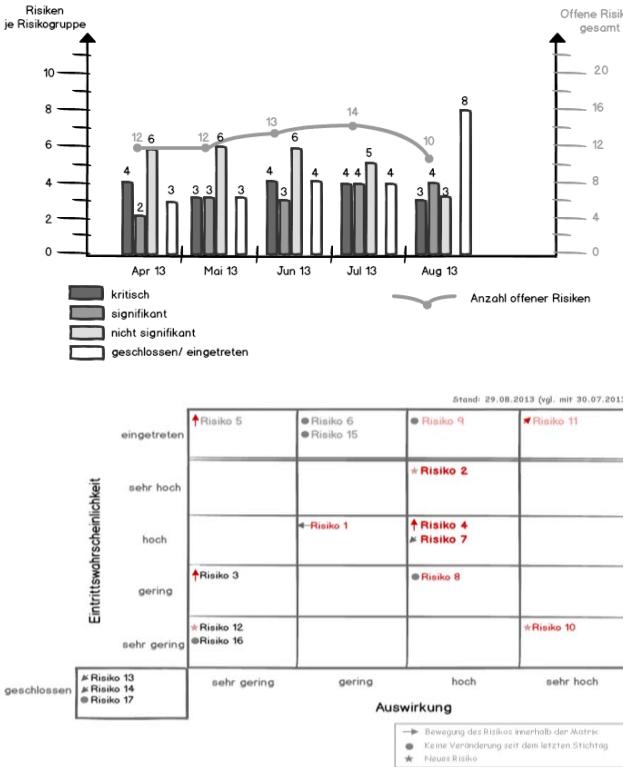

Fig. 7. Prototypes for metric-based risk monitors

We developed prototypes focusing on the visualizations and diagrams that we wanted to address. After just a few iterations we arrived with the central visualizations and diagrams required for the project managers. Two of these final diagrams are shown in figure 7. The first is a classic Cartesian chart with four bar and one line chart showing the number of risks in specific states and the overall number of open risks each on a monthly basis. The second one is an enhancement of a traditional risk matrix. It shows the impact on the bottom and the likelihood on the left, each on a four item scale from very low to very high. The top row contains the risks which did occur and the lower left square contains closed risks. In addition to this we added an icon in front of each risk to indicate how it changed (monthly timing as well).

Project risks were documented using dedicated Excel sheets. These sheets are based on a template which is part of the standard development process and mandatory for the projects. During risk workshops these risk sheets are filled with new risks and existing risks get updated. Additionally, project managers update the risks if something that influences the risk changes (for example a counter measure for the risk is showing to be effective). The risk Excel sheets are stored in CVS repositories.

Handling risks in an Excel sheet is a risk on its own due to lack of consistence checks, constraints, and solid workflow modeling. Hence, it is planned to use Atlassian Jira to model

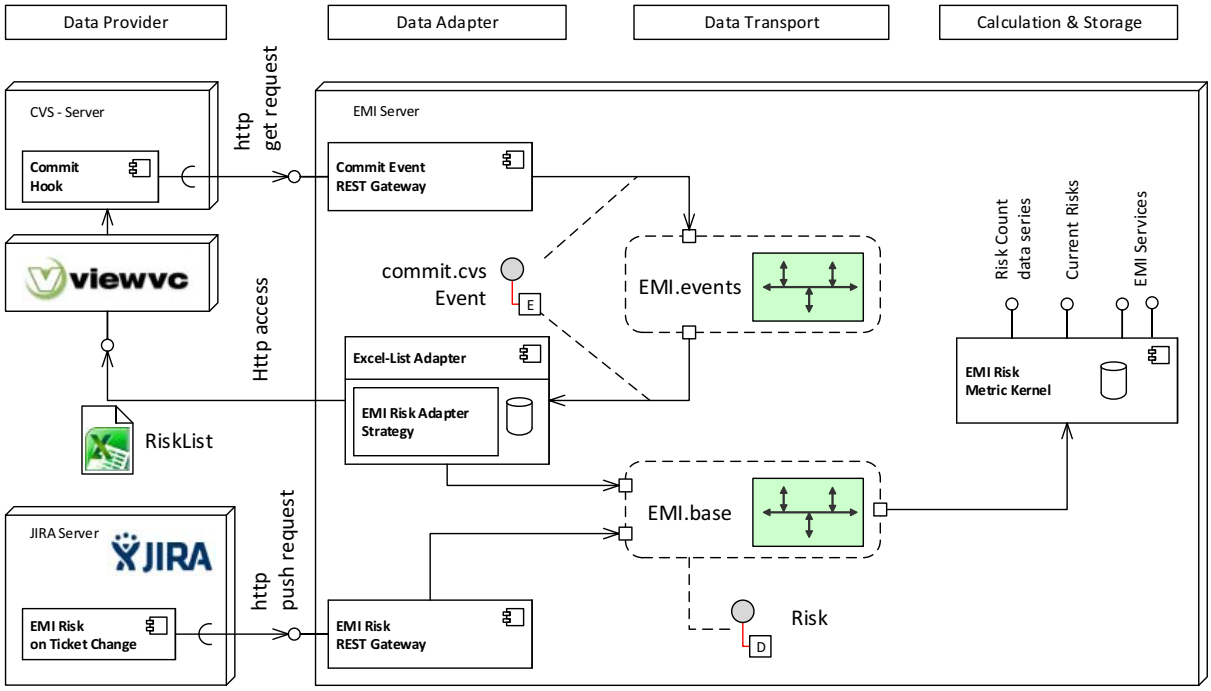

Fig. 8. Static architecture overview of the Data-Adapter (measurement) and Metric-Kernel (calculation) of the EMI for risk-metrics

and store risks in the future. Unfortunately, Jira does not understand the semantic of a risk and hence is not able to provide the risk matrix or bar charts mentioned above. This enforces the need for an independent visualization of risks.

2) Enterprise Measurement Infrastructure

Figure 8 depicts an overview of the static architecture of the developed EMI for the risk metrics. Similar to Figure 6 the view is turned 90 deg. to the right starting with the data providers at the left hand side.

As described above, the risk Excel sheets are stored in a central CVS server. Similar to the SSELab EMI we use a small bash script in a commit hook that again calls the commit event gateway in the EMI on a change of a file. The Excel list data adapter implements the invoke-push adapter pattern using this commit.cvs event. Because the event just contains information about the file that was changed but not the file itself the Excel-List adapter accesses a viewvc server via http to get the specific revision of the Excel file. The data adapter then feeds the Excel file to each strategy that is configured to accept the specific sheet. The risk list adapter strategy analyses the risk list and sends the result, a risk list message, via the base bus of the EMI. An alternative would be an exclusive data adapter just for the risk list. We decided to implement a generic adapter with specific strategies because we anticipated to adapt more Excel sheets in the near future using the same mechanism.

For Jira we again created a small plugin that is executed whenever a risk ticket changes. This plugin then calls the REST API of the risk gateway in the EMI which again creates a risk message on the base bus.

The risk messages on the base bus are received by the risk metric kernel. The kernel first checks some consistency constrains of the incoming risk against the stored risks due to possible data corruption in the Excel sheets (duplicated ids, deleted risks, etc.). Some of these errors can be corrected by the kernel some of them result in a rejection of the incoming message (and error messages to the central logging service as well as indications in the monitoring system). If the risk message passes the check the risk is stored in the data base of the metric kernel. The metrics are calculated on request via the REST APIs of the risk metric kernel.

The EMI was operated on a test stage server hosting the JavaEE implementation of the services as well as all the operation services. Even though this was a test environment we already connected the production stage CVS and viewvc server to test the adaption of risk lists from real projects. The operation was supported by the operation staff of the organization and we again received a lot of positive feedback to the monitoring service as well as system design and composition.

3) Experiences

Again we like to summarize some of the experience that we gained during this project.

- **The reference architecture successfully passed an audit with the architecture management board**
  The CMMI level 3 certified organization required an audit of the reference architecture before we used it inside the organization. The core goal of this is to check the compliance against existing architecture rules and guidelines. The EMI reference architecture was reviewed by several architects and then feed to the architecture management board. It was very well received by all the reviewers as well as the board which lead to the reference architecture passing the audit.

- **Push-Invoke pattern successfully adapts Excel sheets**
  Initially we were struggling with the decision on how to adapt the Excel-based risk lists. We thought about developing dedicated Excel plugins to synchronize the lists simply via a button click. This would require additional clicks when working with risk lists though and we believe that this would lead to some troubles later on. We looked for an alternative and ended up using our invoke push adaption mechanism on the risk lists stored in the CVS version control system. This does not require additional attention from anyone working with the risk list and due to a very open and flexible design of the Excel list adapter it is easy to adapt other types of Excel sheets.

- **Unstructured data requires additional attention**
  Adapting data from a database or other systems like Jira requires very little consistency checks because it is very hard to corrupt the data. Unstructured data like Excel sheets or CSV files, however, require a lot of attention in the metric kernel (and maybe the data adapters) because a lot can (and will!) go wrong. We did an extensive workshop session to discuss possible data corruption scenarios. For some of them we also defined recovery mechanisms (for example a missing row in an excel sheet which can be detected by the metric kernel).

- **Developing, designing and operating an EMI requires training and time**
  The EMI reference architecture provides a very good and structured overview of the different parts of a concrete EMI. Using the reference architecture and developing and operating the specific metric services, however, requires time and training. During this project we trained two metric experts and a few developers and architects and at the end they were able to independently specify and implement an extension to the EMI that uses a pull-forward data adapter and special metric kernels to analyze ticket data from IBM Rational ClearQuest.

V. Conclusion

In this paper we presented the results of the application of the EMI reference architecture in the context of a research infrastructure as well as a large software development organization. It clearly shows how the central roles benefit from the different parts of the reference architecture. We also experienced the benefits of having a dedicated EMI framework to assist with the development of the services. The framework encapsulate the EMDB communication and provides the core operation services and interfaces to them.

Currently we are performing long time field studies to further evaluate and improve the reference architecture. We are also working on a complete set of SDKs and improvement of our EMI framework to support the development of EMIs using javascript and/or JavaEE.

References

[1] C. P. Team, “CMMI® for Development, Version 1.3 CMMI-DEV, V1.3,” 2010.

[2] S. Few, *Information Dashboard Design: The Effective Visual Communication of Data*. O’Reilly Media, Inc., 2006.

[3] M. Kunz, A. Schmietendorf, R. R. Dumke, and C. Wille, “Towards a service-oriented measurement infrastructure,” in *Proc. of the 3rd Software Measurement European Forum (SMEF)*, 2006, pp. 197–207.

[4] S. Architecture, “Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus,” no. April, pp. 1–8, 2006.

[5] D. A. Chappell, *Enterprise service bus*, 1st ed. O’Reilly Media, Inc., 2004.

[6] R. R. Dumke, “Software-Messung und -Bewertung - Eine Bilanz.” 2012.

[7] K. Umapathy, S. Purao, and R. R. Barton, “Designing enterprise integration solutions: effectively,” *Eur. J. Inf. Syst.*, vol. 17, no. 5, pp. 518–527, 2008.

[8] S. Aier and R. Winter, “Fundamental Patterns for Enterprise Integration Services,” *Int. J. Serv. Sci. Manag. Eng. Technol. IJSSMET*, vol. 1, no. 1, pp. 33–49, 2010.

[9] T. Puschmann and R. Alt, “Enterprise Application Integration - The Case of the Robert Bosch Group,” vol. 00, no. c, pp. 1–10, 2001.

[10] R. Dąbrowski, K. Stencel, and G. Timoszuk, “Software is a directed multigraph,” *Softw. Archit.*, pp. 360–369, 2011.

[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)*, 2013.

[12] H. Wache and T. Voegele, “Ontology-based integration of information-a survey of existing approaches,” *IJCAI--01 Work. Ontol. Inf. Shar.*, pp. 108–117, 2001.

[13] P. a. Bernstein and E. Rahm, “A survey of approaches to automatic schema matching,” *VLDB J.*, vol. 10, no. 4, pp. 334–350, Dec. 2001.

[14] J. Lewis and M. Fowler, “Microservices,” 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html. [Accessed: 05-Jul-2014].

[15] H. Kurhinen, “Developing microservice-based distributed workflow engine.” Mikkelin ammattikorkeakoulu, 2014.

[16] Z. MAHMOOD, “Software Products and Technologies for the Development and Implementation of SOA,” *WSEAS Trans. Comput. Res.*, vol. 3, no. 1, pp. 28–34, 2008.

[17] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen, “The Enterprise Service Bus: Making service-oriented architecture real,” *IBM Syst. J.*, vol. 44, no. 4, pp. 781–797, 2005.

[18] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, “Service-Oriented Computing: a Research Roadmap,” *Int. J. Coop. Inf. Syst.*, vol. 17, no. 02, pp. 223–255, Jun. 2008.

[19] R. T. Fielding and R. N. Taylor, “Principled design of the modern Web architecture,” *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[20] A. Steffens, “Entwurf eines Architekturmodells zur Integration heterogener Systeme in MeDIC,” 2013.

[21] M. Vianden, “EMI Homepage,” 2014. [Online]. Available: http://www.enterprise-measurement.com.

[22] C. Herrmann, T. Kurpick, and B. Rumpe, “SSELab: A plug-in-based framework for web-based project portals,” in *TOPI 2012*, 2012, pp. 61–66.

[23] S. Few, *Information Dashboard Design: The Effective Visual Communication of Data*. O’Reilly Media, Inc., 2006.

[24] S. Few, *Show Me the Numbers*. 2012, p. 343.

[25] A. Otto, “Integration einer Metrik-Infrastruktur in die Projektverwaltung SSE-Lab,” 2013.

[26] 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*, 2013, no. CMMI.