Toward Simpler, not Simplistic, Quantification of Software Architecture and Metrics

Report on the Second International Workshop on Software Architecture and Metrics

Ipek Ozkaya, Robert L. Nord
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{ozkaya, rn}@sei.cmu.edu

Heiko Koziolek
ABB Corporate Research
Ladenburg, Germany
heiko.koziolek@de.abb.com

Paris Avgeriou
University of Groningen
Groningen, NL
paris@cs.rug.nl

DOI: 10.1145/2815021.2815037
http://doi.acm.org/10.1145/2815021.2815037

ABSTRACT

Architects of complex software systems face the challenge of how best to assess the achievement of quality attributes and other key system drivers, how to reveal issues and risks early, and how to make decisions about architecture improvement. Software architecture quality has a large impact on this effort, but it is usually not assessed with quantitative measures. A software architecture metric quantifies architecture quality, value, and cost. While it is highly desirable to improve feedback between development and deployment through measurable means for intrinsic quality, value, and cost, efforts in software architecture quality measurement have lagged behind the body of work focusing on code quality. The goal of the Second International Workshop on Software Architecture and Metrics was to discuss progress on architecture and metrics, measurement, and analysis; to gather empirical evidence on the use and effectiveness of metrics; and to identify priorities for a research agenda.

Categories and Subject Descriptors

D.2.8 [**Metrics**], D.2.9 [**Management**], D.2.11 [**Software Architectures**].

General Terms

Design, Management, Measurement.

Keywords

Software architecture, metrics, software analytics, technical debt, software quality, software maintenance and evolution, empirical software engineering, qualitative methods.

1. INTRODUCTION

Software engineers of complex software systems face the challenge of how best to assess the achievement of quality attributes and other key system drivers, how to reveal issues and risks early, and how to make decisions about architecture and system evolution. They increasing need to provide condensed, quantified measurement points for architecture quality. Tracking these measurement points over time can provide insight for managing the pace of software delivery, legacy system evolution, and technology churn. Lack of feedback between development and deployment through measurable means for intrinsic quality, value, and cost has been a key barrier in providing information to assist quantitative and qualitative decision making. The goal of this workshop is to bring together the bodies of work of Software Architecture and Metrics, to bridge this gap.

IEEE1061 defines a software quality metric as *"a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality."* The software engineering community has developed metrics to measure the quality of source code (e.g., size, complexity, coupling, stability). There are also tools for such metrics, such as development environments like Eclipse, Visual Studio, and IDEA or static code analysis tools like Understand, Klocwork, or NDepend. In addition, an active software analytics community mines code repositories (e.g., [1][2][3][4]), issue trackers, and version histories for actionable [5] information.

Despite this substantial body of work focusing on code quality and metrics, its applicability is not proven at the design and architecture levels or at scale. Furthermore, measuring software architecture has received much less attention in research and practice though it is critical to a software system's quality [6]. The most widely used techniques in architecture assessment and decision making rely on expert judgment. We are interested in exploring whether architecture can assist with better contextualizing of existing system and code quality and metrics approaches. Furthermore, we ask whether we need additional architecture-level metrics to make progress in this exploration and whether something as complex and subtle as software architecture can be quantified.

To this end, we initiated the organization of the International Workshop on Software Architecture and Metrics (SAM) series [7]. We proposed a definition for the term *software architecture metric* as follows: "a software quality metric that concerns software architecture and quantifies architecture quality, value, and cost." Different artifacts provide input for computing a software architecture metric: informal architectural documentation, architecture models and views, architecture decisions, source code, byte code, and trace links between an architecture and other artifacts.

The goal of the SAM workshop series is to establish a community that will investigate software architecture and metrics. Furthermore, it aspires to foster discussion on the quality of existing software architecture metrics and create a future research agenda [8][9][10][11][12]. During the first workshop, participants articulated the following challenges [7]:

- How can informal best practices of architecting be codified to derive quantitative metrics?
- Which measures of architecture complexity are most useful so that architects and developers can take actions on appropriate refactorings?
- How can architecture design decisions that are usually captured in textual form, without quantitative quality indicators, be quantitatively assessed for the goodness of architecture decisions?
- How can informal artifacts such as architecture documentation be incorporated into metrics computations to complement the source code, which may not be easy to navigate, to understand tactical decisions?
- How can domain-specific software architecture metrics be defined, and what would their advantage be for certain business

domains (e.g., banking, insurance, avionics, industrial) or technical domains (e.g., embedded, distributed, desktop)?

- When are quantitative metrics more beneficial than qualitative assessments and vice versa?

Potential solutions discussed during the first workshop included creating architecture metrics catalogs, deriving architecture metrics from patterns and styles, establishing a common test bed for architecture metrics, and developing good metrics-computation tools. During the second workshop, which was held in conjunction with the 37th International Conference on Software Engineering (ICSE 2015), presentations focused on driving and categorizing architecture metrics through different architecture approaches such as patterns, styles, and views and developing good tools for communicating the results to different stakeholders, from developers to management decision makers.

The remainder of this paper is structured as follows. Section 2 summarizes presentations of the workshop's participants, illustrating different perspectives on the topic. Section 3 summarizes the discussions that occurred during the workshop by highlighting the potential immediate actions that industry and researchers can take.

2. WORKSHOP CONTRIBUTIONS

The workshop brought together 25 attendees and featured two keynote presentations and nine paper presentations that are available for download [13].

2.1 Keynotes

Kits to Find Bits that Fits (Some Notes on Architecture and Context), presented by Tim Menzies, North Carolina State University. In this presentation, Menzies emphasized that while there is good research in improving the predictive power of software quality metrics, it does not translate well to explain what the metric communicates to the business and other stakeholders. He reflected from his experience that different projects use different metrics for predicting for example defects. Many of the metrics observed at the static code level act differently in different contexts (often contradicting each other), resulting in low external validity and hence low adoption in practice. To be useful, metrics should be studied in their context. Menzies maintained that the actual metrics are not as important per se; it's the discussion that takes place when they are shown to stakeholders. Furthermore, large industries are shifting from measuring source code quality to measuring usage data (e.g., exploring which features are actually used). Software quality and measurement work is based on the assumption that systems and their architectures are hard and costly to change. Menzies argued that as improved development tools and programming become more mainstream, programmers and rework may become cheaper. *If rework were not as costly, would our software architecture and quality metrics challenges and gaps still be relevant? Does it still make sense to come up with universal metrics, or is it more beneficial to derive bottom-up metrics for specific domains and technologies?*

Measuring Software: From Data to Actionable Knowledge, presented by Radu Marinescu, Politehnica University of Timisoara. This keynote reported results from work on building software quality measurement and management tools. Marinescu argued that current tools often risk overemphasizing one aspect of the software under development (to optimize one metric) and compromising another aspect (whose metric was not taken into account). He further observed that to make metrics useful, the fundamental mechanisms need to be abstracted from the developers so that the metrics can be useful to them. He gave as an example the "god class," where expressing the problem of one class doing too much and being tightly coupled communicates the issue to developers much better than presenting the coupling metric of that particular class. Marinescu emphasized that spotting tactical problems is a bigger challenge and suggested encapsulating metrics in rules to make progress. He also warned that through false-positive results, tools can unintentionally introduce confusion and loss of confidence among developers. *How can we make innovative use of software architecture and metrics together encapsulated as rules to detect tactical flaws that slowly diminish the quality of systems? How can we make architecture decisions by looking at system-wide quality attributes and their tradeoffs instead of at individual qualities?*

2.2 Paper Presentations

Paper presentations were limited to seven minutes, followed by a discussion of the work presented, led by a selected discussant.

Metrics for Architectural Synthesis and Evaluation: Use Cases and Compilation by Viewpoint, authored by Olaf Zimmermann, HSR FHO, Switzerland. This work reported on industrial experience in using architecture metrics. Zimmerman explained that architects are motivated to use architecture metrics by the following goals: (a) make and justify architectural decisions, (b) categorize design problems and solutions according to their business context and technical complexity, and (c) compare similar architectures. The metrics that Zimmerman used in practice were mostly sized-based counters, such as number and weight of use cases, number of external interfaces, and number of options considered per problem. Based on these experiences, Zimmerman emphasized that use and relevance of architecture metrics can potentially increase if they are thought through based on the viewpoint of the architecture. For example, measurement approaches applicable when considering the development viewpoint may be different and need to complement applicable physical and process viewpoints.

A Metric-Based Approach to Managing Architecture-Related Impediments in Product Development Flow: An Industry Case Study from Cisco, authored by Ken Power, Cisco Systems, Ireland; and Kieran Conboy, National University of Galway, Ireland. This presentation summarized a case study of how architecture-related impediments impact the flow of work in software engineering teams and organizations. The presentation described how using concepts of flow uncovered early indicators of architectural problems that were impeding creating value for the customer. Focusing on a balance of qualification and quantification using architecture epics, the organization was able to detect when architecture became a cause of interrupted delivery.

Evolution of Object-Oriented Coupling Metrics: A Sampling of 25 Years of Research, authored by Ana Nicolaescu, Horst Lichter, and Yi Xu, RWTH Aachen University, Germany. This presentation reviewed the development and use of coupling metrics and their impact on quality attributes. Thousands of metrics are available in digital libraries, mostly on academic-based systems, but they are accompanied by little evidence. Nicolaescu and colleagues reviewed 26 of the most influential research papers focusing on coupling, complexity, and maintainability. Their analysis revealed that while a very strong theoretical background has been developed, the impact of such coupling and complexity research on practitioners, industry practices, and software analysis tooling is not clear and observable. The authors suggested that the direction of current research should shift toward systematizing and evaluating existing results rather than exploring new applicability domains and defining new metric suites.

Toward Assessing Software Architecture Quality by Exploiting Code Smell Relations, authored by Francesca Arcelli Fontana, University of Milano Bicocca, Italy; Vincenzo Ferme, University of Lugano, Switzerland; and Marco Zanoni, University of Milano Bicocca, Italy. This presentation reported results of evaluating software architecture quality using thresholds as a distribution rather than focusing on single-value results. The presentation emphasized that detecting code or architectural anomalies that give useful hints about possible architecture degradation are more valuable when looking at their co-occurrences. The authors conclude that clusters of code anomalies tend to be better indicators of architectural degradation and maintainability issues than simple metrics evaluation.

An Analysis of Techniques and Methods for Technical Debt Management: A Reflection from the Architecture, authored by Carlos Fernandez-Sanchez, Juan Garbajosa, Carlos Vidal, and Agustin Yague,

Technical University of Madrid, Spain. In this paper, the authors positioned that software architecture and metrics need to be brought together effectively in order to make progress in technical debt management. The presentation summarized a systematic mapping study of available techniques for managing technical debt by highlighting the gaps in software architecture and quantification.

Exploring the Stability of Software with Time-Series Cross-Sectional Data, authored by Jukka Ruohonen, Sami Hyrynsalmi, and Ville Leppänen, University of Turku, Finland. This work investigates stability of software architectures in terms of an object-oriented design principle presented by Robert C. Martin, including abstraction and instability. The authors evaluated the design principle with a time-series cross-sectional regression model. The empirical sample covers a release history from the Java library Vaadin that includes 73 versions and 14 packages. The empirical results establish that the design principle alone could not be used to characterize the library.

Comparing the Applicability of Complexity Measurements for Simulink Models During Integration, authored by Jan Schröder and Christian Berger, University of Gothenburg, Sweden; Thomas Herpel, Automotive Safety Technologies GmbH, Germany; and Miroslaw Staron, University of Gothenburg, Sweden. This work focused on the increase in automotive software by highlighting the growing number of Simulink models for control logic and plant models, which also result in increasing complexity of integration testing and model complexity. The authors evaluated Simulink models from two vehicle projects at a German premium car manufacturer by applying the following three approaches: assessing a model's (a) size, (b) structure, and (c) signal routing. The measurements of 65 models resulted in comparable data for the three measurement approaches. The interviews showed that the expert opinion tends to favor the results of the simple size measurements over the other two.

Architecture-Based Quality Attribute Synergies and Conflicts, authored by Barry Boehm, University of Southern California, United States. This presentation summarized research to develop quality attribute requirement synergies and conflicts matrices that software system engineers can use to identify potential areas of concern in balancing a system's relevant quality attributes. Boehm and colleagues studied key quality attributes that included flexibility, dependability, mission effectiveness, resource utilization, physical capability, cyber capability, and interoperability. These quality attributes represented the top concerns of stakeholders of large, mission-critical systems.

Using Metric Time Lines for Identifying Architecture Shortcomings in Process Execution Architectures, authored by Daniel Lübke, Leibniz Universität Hannover, Germany. This work focused on process execution with service orchestrations for developing business software systems and the challenge of not having business process-related metrics for this architectural style. This presentation described an exploratory study that uses timelines of static process size metrics for constant feedback to software architects that deal with process-oriented architectures. Lübke suggested that by following static code metrics over time, architects can gain a better understanding of how processes and the whole system evolve and whether the metrics provide the expected results as the software evolves.

3. DISCUSSION

Workshop presenters and participants agreed that in order to make progress toward quantification of architecture quality, value, and cost that is relevant to both developers and business stakeholders, effective quantification that combines software architecture and metrics should

- demonstrate useful threshold distributions as opposed to single-value measurements or thresholds
- be supported by software development tools that translate, explain, and contextualize the meaning of the metrics for the stakeholders
- be able to cluster several different quality issues that are relevant rather than focus on isolated issues
- uncover tactical design flaws that slowly degrade the system rather than focus on only execution flaws or defects
- encapsulate design rules
- aim at meaningful discussion among stakeholders instead of blind execution of metric improvement
- avoid one-size-fits-all approaches but focus on individual domains and technologies
- be accompanied by convincing evidence of industrial scale

A major discussion point during the SAM 2015 workshop was the need to bridge the explanation gap for not only business stakeholders but also software developers. Many related code quality metrics do not get adopted due to false-negative results, overhead introduced by tools, and overly complex measurement approaches without immediate value. While using software architecture as an anchor can help avoid such risks, relevance and simple but useful measurement should be the goals.

Finally, the SAM community acknowledges that we do not really have solid theories for measuring architecture quality, value, and cost. This is not necessarily a problem, as we have learned in other software engineering fields. As long as we develop metrics that work and can be used in practice, we can learn valuable lessons. Theories will follow.

3.1 What Problems Are Relevant to Industry?

Some of industry's immediate challenges that have potential relevance to software architecture metrics include the following:

Limited time to compile and interpret the metrics: For a metric to have potential for adoption, it must be easy to use, easy to understand what it is good for, and fully automated. Developers need feedback about the actual impact of rule violations to understand the immediate problem.

Inability to relate design decisions to concrete software architecture artifacts: Architectural metrics are usually difficult to measure because they are related to the decisions that developers make. Metrics related to source code are easier to understand; hence, they dominate the measurements that industry employs despite the fact that their uses are very context specific.

Inability to uncover best practices and success stories: Industry participants emphasized that there is a lot of evidence in industry that if you derive these metrics for software architecture, you obtain a lot of maintenance value. Unfortunately, these stories do not appear at international conferences because they are one-off case studies. Creating opportunities in which to share these case studies would provide invaluable input and avoid wasted time in hypothesis creation and research setup.

3.2 What Metrics Are Needed to Make Reasonable Decisions?

A main theme in both the presentations and discussions was that without simple, yet relevant approaches, making progress in this domain would be hard.

A point measure of a metric may not be meaningful. The trends of metrics, especially a sharp rise or drop in results, can carry more significance than any single result. Time should be spent on defining use cases of potential benefits of combining different metrics and software architecture. Understanding different stakeholders and viewpoints could also potentially help researchers and developers simplify metrics and focus on more beneficial quantification techniques.

Quality attributes must be better utilized. Measurement approaches should focus on the parts of a system that are likely to change. In addition, they should focus on the quality attributes of highest priority. A system that is optimizing for performance may have higher coupling and cohesion values than expected, yet this may not indicate a problem. Design-time qualities, such as "avoid vendor lock-in," are particularly

difficult to quantify. Breaking them down to more tangible, simpler components could be a solution.

Tools need to help interpretation, not only measurement. Metrics should be used to understand the system, not just to assess the system. They should relate to both value and cost and should help interpret the results. Despite years of code quality work, executives do not fully understand what they are getting in the delivered software, which is a risk.

Role-based architectural metrics. Different target audiences may find more meaning in different collections of architectural metrics. A CIO has one view of a software system and its architectural metrics and a developer responsible for a particular subsystem has another view. Current tools for metrics calculation often assume a single stakeholder role as the target audience, which can lead to confusion.

4. CONCLUSIONS

Given the broad participation by both academics and industrial practitioners and the intense discussions that were held, we believe there is interest and momentum to establish a substantial stream of work in this area. Work presented at this second SAM workshop represented a healthy cross-fertilization of software analysis tools, software architecture and design decision making, and code quality. We already consider that the identified research challenges and the solution approaches are a great starting point for the research community, best driven forward in close collaboration with the industry. Finally, we believe in the cross-fertilization between the SAM community and the other communities related to metrics, software analytics, and mining software repositories. We are amenable to organizing future editions of the workshop and supporting the SAM community to grow and flourish.

5. ACKNOWLEDGMENTS

Putting together SAM 2015 was a team effort. We thank the authors of all submitted papers and invited speakers for providing the content of the program. We would like to express our gratitude to the program committee, who contributed their expertise, provided valuable feedback to the authors, and helped improve the quality of the accepted papers:

- Pierre America, Philips Research, NL
- Ayse Bener, Ryerson University, CA
- Barry Boehm, University of Southern California, US
- Eric Bouwers, Technical University Delft, NL
- Yuangfang Cai, Drexel University, US
- Jane Cleland-Huang, DePaul University, US
- Rich Hilliard, Consulting Software Systems Architect, US
- Oliver Hummel, iQser, DE
- Anton Jansen, ABB, SE
- Rainer Koschke, University of Bremen, DE
- Philippe Kruchten, University of British Columbia, CA
- Patricia Lago, VU University, NL
- Nazim Madhavji, University of Western Ontario, CA
- Radu Marinescu, “Politehnica” University of Timisoara, RO
- Tim Menzies, North Carolina State University, US
- Matthias Naab, Fraunhofer, DE
- Oscar Pastor, Valencia University of Technology, ES
- Neeraj Sangal, Lattix, US
- Jean-Guy Schneider, Swinburne University of Technology, AU
- Bran Selic, Malina Software Corp., CA
- Will Snipes, ABB, US
- Michael Stal, Siemens, DE
- Robert Stoddard, Software Engineering Institute, US
- Uwe Zdun, University of Vienna, AT
- Liming Zhu, National ICT Australia, AU
- Olaf Zimmermann, University of Applied Sciences, CH
- Tom Zimmermann, Microsoft Research, USA

6. DISCLAIMER

7. REFERENCES

[1] The Working Conference on Mining Software Repositories. http://www.msrconf.org/.

[2] Menzies, T. and Zimmermann, T. 2013. Software analytics: so what? *IEEE Software* 30, 4 (Jul./Aug. 2013), 31-37.

[3] Xie, T., Zimmermann, T., and van Deursen, A. 2013. Introduction to the special issue on mining software repositories. *Empir. Softw. Eng.* 18, 6 (Dec. 2013), 1043-1046.

[4] Workshop on Emerging Trends in Software Metrics (WeTSOM). http://www.rcost.unisannio.it/wetsom2014/#previouseditions.

[5] Zimmermann, T., Weißgerber, P., Diehl, S., and Zeller, A. 2004. Mining version histories to guide software changes. In *Proceedings of the 26th International Conference on Software Engineering* (Edinburgh, UK, May 23-28, 2004). IEEE Computer Society, Washington, DC, 563-572.

[6] Avgeriou, P., Stal, M., and Hilliard, R. 2013. Architecture sustainability. *IEEE Software* 30, 6 (Nov./Dec. 2013), 40-44.

[7] Nord, R. L., Ozkaya, I., Koziolek, H., and Avgeriou, P. 2014. Quantifying software architecture quality report on the First International Workshop on Software Architecture Metrics. *ACM SIGSOFT* 39, 5 (Sep. 2014), 4:32-4:34.

[8] Koziolek, H. 2011. Sustainability evaluation of software architectures: a systematic review. In *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems* (Boulder, CO, Jun. 20-24, 2011). ISARCS, New York, NY, 3-12.

[9] Kruchten, P., Nord, R., and Ozkaya, I., Eds. 2012. *IEEE Software*, Special Issue on Technical Debt, 29, 6 (Nov./Dec. 2012).

[10] Callo Arias, T., van der Spek, P., and Avgeriou, P. 2011. A practice-driven systematic review of dependency analysis solutions. *Empir. Softw. Eng.* 16, 5 (Oct. 2011), 1-43.

[11] Macia, I., Garcia, J., Popescu, D., Garcia, A., Medvidovic, N., and von Staa, V. 2012. Are automatically-detected code anomalies relevant to architectural modularity? An exploratory analysis of evolving systems. In *Proceedings of the 11th Annual International Conference on Aspect-Oriented Software Development* (Potsdam, Germany, Mar. 25-30, 2012). ACM, New York, NY, 167-178.

[12] Bouwers, E., van Deursen, A., and Visser, J. 2013. Evaluating usefulness of software metrics: an industrial experience report. In *Proceedings of the 2013 International Conference on Software Engineering* (San Francisco, CA, May 18-26, 2013). IEEE Press, Piscataway, NJ, 921-930.

[13] Second International Workshop on Software Architecture Metrics. http://www.sei.cmu.edu/community/sam2015/.