

# Software Development Analytics

Edited by

Harald Gall<sup>1</sup>, Tim Menzies<sup>2</sup>, Laurie Williams<sup>3</sup>, and  
Thomas Zimmermann<sup>4</sup>

- 1 Universität Zürich, CH, [gall@ifi.uzh.ch](mailto:gall@ifi.uzh.ch)
- 2 North Carolina State University, US, [tim@menzies.us](mailto:tim@menzies.us)
- 3 North Carolina State University, US, [williams@csc.ncsu.edu](mailto:williams@csc.ncsu.edu)
- 4 Microsoft Research – Redmond, US, [tzimmer@microsoft.com](mailto:tzimmer@microsoft.com)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 14261 “Software Development Analytics”. We briefly summarize the goals and format of the seminar, the results of the break out groups, and a draft of a manifesto for software analytics. The report also includes the abstracts of the talks presented at the seminar.

**Seminar** June 22–27, 2014 – <http://www.dagstuhl.de/14261>

**1998 ACM Subject Classification** D.2 Software Engineering

**Keywords and phrases** software development, data-driven decision making, analytics, empirical software engineering, mining software repositories, business intelligence, predictive analytics

**Digital Object Identifier** 10.4230/DagRep.4.6.64

**Edited in cooperation with** Reid Holmes (University of Waterloo, CA)

## 1 Executive Summary

*Harald Gall*

*Tim Menzies*

*Laurie Williams*

*Thomas Zimmermann*

**License**  Creative Commons BY 3.0 Unported license  
© Harald Gall, Tim Menzies, Laurie Williams, and Thomas Zimmermann

Software and its development generate an inordinate amount of data. For example, check-ins, work items, bug reports and test executions are recorded in software repositories such as CVS, Subversion, GIT, and Bugzilla. Telemetry data, run-time traces, and log files reflect how customers experience software, which includes application and feature usage and exposes performance and reliability. The sheer amount is truly impressive:

- As of July 2013, Mozilla Firefox had 900,000 bug reports, and platforms such as Sourceforge.net and GitHub hosted millions of projects with millions of users.
- Industrial projects have many sources of data at similar scale.

But how can this data be used to improve software? Software analytics takes this data and turns it into actionable insight to inform better decisions related to software. Analytics is commonly used in many businesses—notably in marketing, to better reach and understand customers. The application of analytics to software data is becoming more popular.

To a large extent, software analytics is about what we can learn and share about software. The data include our own projects but also the software projects by others. Looking back



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Software Development Analytics, *Dagstuhl Reports*, Vol. 4, Issue 6, pp. 64–83

Editors: Harald Gall, Tim Menzies, Laurie Williams, and Thomas Zimmermann



DAGSTUHL  
REPORTS

Dagstuhl Reports  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

at decades of research in empirical software engineering and mining software repositories, software analytics lets us share all of the following:

- **Sharing insights.** Specific lessons learned or empirical findings. An example is that in Windows Vista it was possible to build high-quality software using distributed teams if the management is structured around code functionality (Christian Bird and his colleagues).
- **Sharing models.** One of the early models was proposed by Fumio Akiyama and says that we should expect over a dozen bugs per 1,000 lines of code. In addition to defect models, plenty of other models (for example effort estimation, retention and engagement) can be built for software.
- **Sharing methods.** Empirical findings such as insights and models are often context-specific, e. g., depend on the project that was studied. However, the method (“recipe”) to create findings can often be applied across projects. We refer to “*methods*” as the techniques by which we can transform data into insight and models.
- **Sharing data.** By sharing data, we can use and evolve methods to create better insight and models.

The goal of this seminar was to build a roadmap for future work in this area. Despite many achievements, there are several challenges ahead for software analytics:

- How can we make data useful to a wide audience, not just to developers but to anyone involved in software?
- What can we learn from the vast amount of unexplored data?
- How can we learn from incomplete or biased data?
- How can we better tie usage analytics to development analytics?
- When and what lessons can we take from one project and apply to another?
- How can we establish smart data science as a discipline in software engineering practice and research as well as education?

## Seminar Format

In this seminar, we brought together researchers and practitioners from academia and industry who are interested in empirical software engineering and mining software repositories to share their insights, models, methods, and/or data. Before the seminar, we collected input from the participants through an online survey to collect relevant themes and papers for the seminar. Most themes from the survey fell into the categories of method (e. g., measurement, visualization, combination of qualitative with quantitative methods), data (e. g., usage/telemetry, security, code, people, etc.), and best practices and fallacies (e. g., how to choose techniques, how to deal with noise and missing data, correlation vs. causation). A theme that also emerged in the pre-Dagstuhl survey was analytics for the purpose of theory format, i. e., “*data analysis to support software engineering theory formation (or, data analytics in support of software science, as opposed to software engineering)*”.

At the seminar, we required that attendees

1. discuss the next generation of software analytics;
2. contribute to a *Software Analytics Manifesto* that describes the extent to which software data can be exploited to support decisions related to development and usage of software.

Attendees were required to outline a set of challenges for analytics on software data, which will help to focus the research effort in this field. The seminar provided ample opportunities for discussion between attendees and also provide a platform for collaboration between attendees since our time was divided equally between:

1. Plenary sessions where everyone gave short (10 minute) presentations on their work.
2. Breakout sessions where focus groups worked on shared tasks.

Our schedule was very dynamic. Each day ended with a “think-pair-share” session where some focus for the next day was debated first in pairs, then shared with the whole group. Each night, the seminar organizers would take away the cards generated in the “think-pair-share” sessions and use that feedback to reflect on how to adjust the next day’s effort.

## 2 Table of Contents

### Executive Summary

*Harald Gall, Tim Menzies, Laurie Williams, and Thomas Zimmermann* . . . . . 64

### The Manifesto

Statements defining data science and analytics . . . . . 69

General statements . . . . . 69

Statements about people . . . . . 69

Statements about data . . . . . 70

Statements about methods . . . . . 70

Statements about results and outcomes . . . . . 70

**Follow-up Work** . . . . . 70

**Overview of Talks** . . . . . 71

Emotion Mining for Software Developers  
*Bram Adams* . . . . . 71

Software analytics with email data  
*Alberto Bacchelli* . . . . . 71

Software Analytics to Build Recommender Systems  
*Ayse Bener* . . . . . 72

Data-Driven Engineering at Microsoft  
*Trevor Carnahan* . . . . . 72

Misconceptions about mining bug repositories  
*Serge Demeyer* . . . . . 72

Rationalism vs. Empiricism in Software Engineering  
*Premkumar T. Devanbu* . . . . . 73

A tale of two datasets  
*Georgios Gousios* . . . . . 73

The trouble with performance analytics  
*Abram Hindle* . . . . . 73

Applying Qualitative Analytics  
*Reid Holmes* . . . . . 74

Information > Tool or Information → Tool?  
*Miryung Kim* . . . . . 74

Thoughts on selling software analytics to software companies  
*Andrew J. Ko* . . . . . 75

An ARCADE for Architecture Analytics  
*Nenad Medvidovic* . . . . . 75

Software Effort Estimation Models – Past, Present and Future  
*Leandro L. Minku* . . . . . 75

Operational Data are not Experimental Data <i>Audris Mockus</i> . . . . .	76
Analytics on Ad Library Maintenance in Android Apps <i>Meiyappan Nagappan</i> . . . . .	77
Are We Really Helping Developers? <i>Alessandro Orso</i> . . . . .	77
My flings with data analysis <i>Venkatesh-Prasad Ranganath</i> . . . . .	78
Towards the Impact of Software Analytics <i>Guenther Ruhe</i> . . . . .	78
Mere Numbers aren't Enough – Focus on Interpretation and Visualization <i>Per Runeson</i> . . . . .	78
Composable Data Mining: Supporting Analytics for End Users <i>Anita Sarma</i> . . . . .	79
42 years of Unix history in one repository <i>Diomidis Spinellis</i> . . . . .	79
Open Problems and Challenges in Software Analytics <i>Diomidis Spinellis</i> . . . . .	80
Studying social media in software development: reflecting on research methods <i>Margaret-Anne Storey</i> . . . . .	80
The Graph <i>Burak Turhan</i> . . . . .	81
Why Quality Models Don't Work and Why We Need Them Anyway <i>Stefan Wagner</i> . . . . .	81
Analytics in the Field <i>Patrick Wagstrom</i> . . . . .	81
Software Analytics in Practice <i>Dongmei Zhang</i> . . . . .	82
<b>Breakout Groups</b> . . . . .	82
<b>Participants</b> . . . . .	83

### 3 The Manifesto

To compile the manifesto, we followed three steps:

1. In a think-pair-share session we collected 150 statements that participants felt should be part of the manifesto. Over the Dagstuhl week, the organizers sorted the cards into categories.
2. A breakout group compiled a draft manifesto, which was then used in a plenary session to establish core groups that should be part of a manifesto. The resulting groups were definitions, general statements, people, data, methods, results and outcomes.
3. After the seminar, the organizers selected representative statements, which were then rated by 22 attendees as part of a post-Dagstuhl survey (“In your opinion, how important is it to include this statement for a manifesto on data science in software engineering?”).

In the rest of this section, we list the statements that were rated favorably by 66.6% of the survey participants as *Essential* (*E*) or *Worthwhile* (*W*). Statements that were rated by 40.0% of survey participants as *Essential* are printed in bold.

#### 3.1 Statements defining data science and analytics

- **Software analytics is to utilize data-driven approaches to obtain insightful and actionable information to help software practitioners with their data related tasks** (E: 0.682, W: 0.136)
- Data science in SE should lead to: (one of) Insights about users; Advise for practitioners (which tools to use, design); Theory for researchers; Innovations for all (E: 0.364, W: 0.318)

#### 3.2 General statements

- **Your project has a history. Learn from it. Decide from it. Embrace it.** (E: 0.429, W: 0.381)
- **What counts is insights not numbers!** (E: 0.429, W: 0.381)
- **Exploration matters.** (E: 0.4, W: 0.4)
- We strive to do the best we can with the evidence at hand, but we accept that that evidence may be incomplete, noisy, and even wrong (E: 0.364, W: 0.455)
- We will be able to gain insights from the past to improve the future. (E: 0.333, W: 0.381)
- Data, analyses, methods and results have to be publicly shared (E: 0.227, W: 0.455)
- SE data science should be actionable, reproducible. Should not be about finding a way to apply your hammer but finding solutions to real problem. (E: 0.19, W: 0.524)
- Good data science does not get in the way of developing software (distraction, additional data collection) but supports it (makes it more efficient) (E: 0.143, W: 0.571)
- Measure before action; act; measure again. (E: 0.136, W: 0.545)
- Generalizations should be viewed with a healthy skepticism (E: 0.095, W: 0.571)

#### 3.3 Statements about people

- **Data doesn't decide, people do.** (E: 0.545, W: 0.136)
- Good data science takes the people into the account not just code and checkins (aka, in a study, ask the people involved if your results make sense) (E: 0.364, W: 0.5)

### 3.4 Statements about data

- Understand where the data comes from, accepting that it may be gamed (E: 0.5, W: 0.318)
- **Quality of data is more than quantity** (E: 0.409, W: 0.409)
- **Impact requires actionable data** (E: 0.4, W: 0.2)
- Data is merely one component of the large amount of insight, experience, and knowledge that informs decisions (E: 0.318, W: 0.409)
- Do check your data multiple times (E: 0.227, W: 0.591)

### 3.5 Statements about methods

- Engage domain experts in validation of analysis (E: 0.571, W: 0.381)
- **Interpretation and visualization is central to data science** (E: 0.5, W: 0.364)
- **We value qualitative study as much as quantitative study; often that's where the insights come from** (E: 0.476, W: 0.429)
- **Replicate and triangulate** (E: 0.455, W: 0.364)
- Big data research should not only consider machine generated data. Qualitative data are of equal importance. (E: 0.381, W: 0.381)
- Effect size matters (E: 0.35, W: 0.45)
- Actionable impact over sophisticated method. (E: 0.333, W: 0.429)
- When it comes to metrics, more is not necessarily better. (E: 0.286, W: 0.381)
- Context must accompany every method. (E: 0.238, W: 0.571)
- Data Science is integrating and analyzing data from different sources. (E: 0.19, W: 0.524)

### 3.6 Statements about results and outcomes

- **Communicating results is as important as computing results** (E: 0.524, W: 0.333)
- **Analytics should lead to action** (E: 0.476, W: 0.286)
- Make results actionable and relevant (E: 0.381, W: 0.429)
- Publish what didn't work. (E: 0.364, W: 0.545)
- Data science should produce actionable findings (E: 0.333, W: 0.476)
- Value usefulness to decide over precision or correctness (E: 0.318, W: 0.682)

## 4 Follow-up Work

At the seminar, it was recognized that the community needs a web portal to store and distribute its community product. That portal is currently being developed.

Also, attendees commented there were many “best practices” that were unknown to the broader community. This resulted in an all-too-varied performance result when newcomers struggled to apply analytics to their particular project. Hence, it was decided to co-write a book “**Perspectives on Data Science for Software Engineering**” where each (small) chapter would be written by one Dagstuhl seminar attendee as well as other well-known people in the field.

## 5 Overview of Talks

### 5.1 Emotion Mining for Software Developers

*Bram Adams (Polytechnique Montreal, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Bram Adams

**Joint work of** Murgia, Alessandro; Tourani, Parastou; Adams, Bram; Ortu, Marco

**Main reference** A. Murgia, P. Tourani, B. Adams, M. Ortu, “Do Developers Feel Emotions? An Exploratory Analysis of Emotions in Software Artifacts,” in Proc. of the 11th Working Conf. on Mining Software Repositories (MSR’14), pp. 262–271, ACM, 2014.

**URL** <http://dx.doi.org/10.1145/2597073.2597086>

Software development is a collaborative activity in which developers interact to create and maintain a complex software system. Human collaboration inevitably evokes emotions like joy or sadness, which can affect the collaboration either positively or negatively, yet not much is known about the individual emotions and their role for software development stakeholders. We analyzed whether development artifacts like issue reports carry any emotional information about software development. This is a first step towards verifying the feasibility of an automatic tool for emotion mining in software development artifacts: if humans cannot determine any emotion from a software artifact, neither can a tool. Analysis of the Apache Software Foundation issue tracking system shows that developers do express emotions (in particular gratitude, joy and sadness), yet more investigation is needed before building a fully automatic emotion mining tool.

### 5.2 Software analytics with email data

*Alberto Bacchelli (TU Delft, NL)*

**License** © Creative Commons BY 3.0 Unported license  
© Alberto Bacchelli

**Joint work of** Bacchelli, Alberto; Lanza, Michele; Humpa, Viteszlav

**Main reference** A. Bacchelli, M. Lanza, V. Humpa, “RTFM (Read the Factual Mails) – Augmenting Program Comprehension with Remail,” in Proc. of the 15th European Conf. on Software Maintenance and Reengineering (CSMR’11), pp. 15–24, IEEE, 2011; pre-print available from author’s webpage.

**URL** <http://dx.doi.org/10.1109/CSMR.2011.6>

**URL** <http://sback.it/publications/csmr2011.pdf>


The evolution of software systems leaves its traces in a number of artifacts. Some artifacts are made of structured data (e.g., source code) that is easily parseable, while others are made of unstructured data (e.g., documentation and emails) that is more difficult to analyze. Nevertheless unstructured data contain precious knowledge to support software engineering.

In this talk I provide initial evidence that email data can be effectively used as a valuable target for software analytics. In particular, I will present anecdotal evidence on the usefulness of email data in supporting four program comprehension tasks, namely (1) finding Entry Points in a software system, (2) conducting Software Evolution Analysis, (3) improving Expert Finding techniques, and (4) recovering Additional Documentation about system’s entities. The aim of the presentation is to trigger future collaboration and project in using unstructured software data to support software engineering.



### 5.3 Software Analytics to Build Recommender Systems

*Ayse Bener (Ryerson University – Toronto, CA)*

License  Creative Commons BY 3.0 Unported license  
© Ayse Bener

The goal of evidence based analytics for software systems is to create methods, techniques, tools and processes to improve processes and/or to efficiently allocate resources. We need to migrate from prediction to recommendation by building models that focus on reasons and casual relationships. We need to integrate prediction models into business rules and processes to improve software development processes as well as modeling people aspects. There are also other challenges that both research and practice should consider. One of these challenges is reproducibility of approaches since no one shares enough data and there are no standard process/ framework to conduct analytics. Another challenge is the lack of integration of research tools into development platforms such as Jira, GitHub, Eclipse, etc.

### 5.4 Data-Driven Engineering at Microsoft

*Trevor Carnahan (Microsoft Research – Redmond, US)*

License  Creative Commons BY 3.0 Unported license  
© Trevor Carnahan

An inside look into Microsoft efforts to apply analytics into data-driven engineering of software and a fresh look at a modern developer's responsibilities and activities. This talk starts with general decision making data concepts at work. Then the talk outlines a few data systems developed in SQL Server and in Tools for Software Engineers currently in use for decisions. It finishes with the impact of devops and more service and cloud development to challenge conceptions of a software engineer.

### 5.5 Misconceptions about mining bug repositories

*Serge Demeyer (University of Antwerp, BE)*

License  Creative Commons BY 3.0 Unported license  
© Serge Demeyer

In this talk I present a few misconceptions that some researchers have about the records maintained in software repositories in general and bug repositories in particular. These misconceptions were harvested from a series of focus groups we organised with industrial team leads about promising research mature enough to be applied in practice.

In no particular order (and without criticising researchers working on these problems) we received the following feedback.

- Recommenders for the component of a particular bug are more useful than recommenders for the best person to fix.
- The time to fix is often seen as the time between opening and closing a bug. A more actionable definition is the time between a developer acknowledging that he will fix a bug and the time he submits it for review.

- Accuracy (precision / recall in all its variations) is not the primary criterion to be optimised. Equally important is the learning curve, i. e. how many bug reports you need for training before the recommender achieves a reasonable accuracy.

## 5.6 Rationalism vs. Empiricism in Software Engineering

*Premkumar T. Devanbu (University of California – Davis, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Premkumar T. Devanbu

Researchers in software engineering have each subscribed almost exclusively to one of two approaches: Rationalist, and Empiricist. This talk is a plea for more overlap and interaction, specially from the latter to the former.

## 5.7 A tale of two datasets

*Georgios Gousios (TU Delft, NL)*

**License** © Creative Commons BY 3.0 Unported license  
© Georgios Gousios  
**Main reference** G. Gousios, “The GHTorrent dataset and tool suite,” in Proc. of the 10th Working Conference on Mining Software Repositories (MSR’13), pp. 233–236, IEEE/AMC, 2013.  
**URL** <http://dl.acm.org/citation.cfm?id=2487085.2487132>

What drives reuse of shared research artifacts? In my talk, I argue that the openness of the construction process plays a central role in the dissemination of research artifacts and their acceptance and trust by other researchers.

## 5.8 The trouble with performance analytics

*Abram Hindle (University of Alberta, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Abram Hindle  
**URL** <http://softwareprocess.es>

Performance Analytics are continually challenged by the lack performance information within existing software repositories. The lack of logging by developers and users leads to this problem. Until developers are motivated by better analytics and tools, they will not be motivated to log performance information and aggregate it. It is up to us as researchers to generate this data, demonstrate how great the tools and analytics are with this information, and then motivate developers to log this information in the hope of using our methods and tools.

## 5.9 Applying Qualitative Analytics

*Reid Holmes (University of Waterloo, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Reid Holmes

**Joint work of** Baysal Olga; Holmes, Reid; Godfrey, Mike

**Main reference** O. Baysal, R. Holmes, M. Godfrey, “No Issue Left Behind: Reducing Information Overload in Issue Tracking,” in Proc. of the 22nd ACM SIGSOFT Int’l Symposium on the Foundations of Software Engineering (FSE’14), 2014, to appear.

Modern software development processes generate large amounts of metadata. While it is tempting to aggregate this data in forms that are amenable to graphical representations (e. g., charts of defects fixed over time), these representations are not useful for developers as they address their day-to-day tasks.

This talk describes a research project that examined the kinds of questions industrial developers want to answer using modern issue tracking systems along with other questions they would like to ask but are unable to ask using existing tools. Ultimately we find that developers want support for expressing queries for addressing specific tasks along with maintaining overall situational awareness of both their own issues along with other issues that are relevant to their interests. We have created a model of these information needs and have built a prototype tool called Dash that fulfills these shortcomings by providing a qualitative (rather than quantitative) projection of issue tracker data. We have iterated on this tool several times with industrial developers and it has currently been deployed within Mozilla so we can gather longitudinal usage data to validate and improve our information model.

## 5.10 Information > Tool or Information → Tool?

*Miryung Kim (University of Texas – Austin, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Miryung Kim

**Main reference** M. Kim, T. Zimmermann, N. Nagappan, “An Empirical Study of Refactoring Challenges and Benefits at Microsoft,” *Trans. on Software Engineering*, 40(7):633–649, 2014.

**URL** <http://dx.doi.org/10.1109/TSE.2014.2318734>

In this talk, I present my work on systematic changes with the aim of having a discussion on whether information produced by development analytics is more important than building development tools or whether we should think about deriving information that could inform the design of development tools.

### References

- 1 Kim, M.; Zimmermann, T.; Nagappan, N., “An Empirical Study of Refactoring Challenges and Benefits at Microsoft,” *IEEE Transactions on Software Engineering*, 40(7):633–649, DOI: 10.1109/TSE.2014.2318734.

## 5.11 Thoughts on selling software analytics to software companies

*Andrew J. Ko (University of Washington – Seattle, US)*

License © Creative Commons BY 3.0 Unported license  
© Andrew J. Ko

I co-founded a company that provides contextual crowdsourced help to SaaS companies, providing a stream of insights about the questions, confusions, and struggles that users are having on their site. In interacting with customers, I have found that very few customers have mature practices or understanding of metrics, measurement, hypotheses, or experiments. This heavily skews what can be sold, how customers perceive value, and what types of analytics they understand and desire.

## 5.12 An ARCADE for Architecture Analytics

*Nenad Medvidovic (University of Southern California, US)*

License © Creative Commons BY 3.0 Unported license  
© Nenad Medvidovic

From its very inception, the study of software architecture has recognized architectural decay as a regularly-occurring phenomenon in long-lived systems. At the same time, there is a relative dearth of empirical data about the nature of architectural change and the actual extent of decay in existing systems. In this talk, I present a workbench developed to help us take a step toward addressing that scarcity, by automating the study of architectural change and decay. The workbench, ARCADE (“Architecture Recovery, Change, and Decay Evaluator”) enabled us to conduct a pilot study of the evolution of software architectures from several hundred versions belonging to 12 open- source systems totalling over 112 million source lines of code. To lay the groundwork for ARCADE, we previously performed an extensive evaluation of state-of-the-art techniques for obtaining a software architecture from a system’s implementation and (2) cataloged symptoms of architectural decay. This talk reported on several results from the study, which revealed a number of unexpected findings regarding the frequency of architectural changes in software systems, the rate at which architectural decay occurs, the nature of architectural decay, and its relationship to code-level decay.

## 5.13 Software Effort Estimation Models – Past, Present and Future

*Leandro L. Minku (University of Birmingham, GB)*

License © Creative Commons BY 3.0 Unported license  
© Leandro L. Minku

In this talk, I briefly go through some key points in terms of the past, present and future of software effort estimation models. I go from (1) conclusion instability to (2) ensembles [1, 3, 4] and locality [2, 3] to (3) the importance of concentrating more on temporal [5] and cross-company learning [5, 6], generating insights [5], and obtaining a better understanding of when, why and how our models work (or don’t work). Even though the talk is in the

context of software effort estimation models, several of these ideas are also applicable to other types of software prediction models, such as defect predictors.

### References

- 1 L.L. Minku and X. Yao. Software Effort Estimation as a Multi- objective Learning Problem, 22(4):35, ACM TOSEM 2013.
- 2 T. Menzies et al. Local versus Global Lessons for Defect Prediction and Effort Estimation. 39(6):822–834, IEEE TSE 2013.
- 3 L.L. Minku and X. Yao. Ensembles and Locality: Insight on Improving Software Effort Estimation, 55(8):1512–1528, IST 2013.
- 4 Y. Kultur, B. Turhan and A. Bener. Ensemble of Neural Networks with Associative Memory (ENNA) for Estimating Software Development Costs, 22(6):395–402, KBS 2009.
- 5 L.L. Minku and X. Yao. How to Make Best Use of Cross- company Data in Software Effort Estimation?, pp. 446–456, ICSE 2014.
- 6 E. Kocaguneli, T. Menzies and E. Mendes. Transfer Learning in Effort Estimation. ESE 2014 (in press).

## 5.14 Operational Data are not Experimental Data

*Audris Mockus (Avaya – Basking Ridge, US)*

License  Creative Commons BY 3.0 Unported license  
© Audris Mockus

**Main reference** A. Mockus, “Engineering big data solutions,” in Proc. of the “Future of Software Engineering” (FOSE) track at the 36th Int’l Conf. on Software Engineering (ICSE’14), pp. 85–99, ACM, 2014; pre-print available at author’s webpage.

**URL** <http://dx.doi.org/10.1145/2593882.2593889>

**URL** <http://mockus.org/papers/BigData.pdf>

The collection and use of low-veracity data in software repositories and other operational support systems is exploding. It is, therefore, imperative to elucidate basic principles of how such data comes into being and what it means. Are there practices of constructing software data analysis tools that could raise the integrity of their results despite the problematic nature of the underlying data? The talk explores the basic nature of data in operational support systems and considers approaches to develop engineering practices for software mining tools.

### References

- 1 Audris Mockus. Engineering big data solutions. In *ICSE’14 FOSE*, 2014.
- 2 Audris Mockus. Missing data in software engineering. In J. Singer et al., editor, *Guide to Advanced Empirical Software Engineering*, pages 185–200. Springer-Verlag, 2008.
- 3 Audris Mockus. Software support tools and experimental work. In V Basili and et al., editors, *Empirical Software Engineering Issues: Critical Assessments and Future Directions*, volume LNCS 4336, pages 91–99. Springer, 2007.

## 5.15 Analytics on Ad Library Maintenance in Android Apps

*Meiyappan Nagappan (Rochester Institute of Technology, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Meiyappan Nagappan

**Main reference** I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, A. E. Hassan, “On the Relationship between the Number of Ad Libraries in an Android App and its Rating,” *IEEE Software*, published online, 1 page, 2014.

**URL** <http://dx.doi.org/10.1109/MS.2014.79>

With more than 75% of mobile apps today being free-to-download, advertisement within apps is one of the key business models to generate revenue. Advertisements are served through the embedding of specialized code, i. e., ad libraries. Unlike other types of libraries, developers cannot ignore new versions of the embedded ad libraries or new ad libraries without risking a loss in revenue. However, updating ad libraries also has expenses, which can become a major problem as ad library updates are becoming more prevalent in mobile apps.

We mined over 128,000 Android apps over 12 months. After removing apps that were considered as noise, an analysis of 13,983 versions of 5,937 Android apps shows that almost half (48.98%) of the studied versions had an ad library update (i. e., ad library was added, removed, or updated). Interestingly, in 13.75% of app updates (new version in the Google Play store) with at least one case of ad library update, we found no changes to the app’s own API, which suggests substantial additional effort for developers to maintain ad libraries. We also explore the rationales for why such updates are carried out. Finally, we find no evidence that the number of ad libraries in an app is related to the ratings that an app can get. However, integrating certain specific ad libraries can negatively impact the rating of an app.

## 5.16 Are We Really Helping Developers?

*Alessandro Orso (Georgia Institute of Technology, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Alessandro Orso

**Joint work of** Parnin, Chris; Orso, Alessandro

**Main reference** C. Parnin, A. Orso, “Are Automated Debugging Techniques Actually Helping Programmers?” in *Proc. of the Int’l Symp. on Software Testing and Analysis (ISSTA’11)*, pp. 199–209, ACM, 2011.

**URL** <http://dx.doi.org/10.1145/2001420.2001445>

This talk discusses some of the risks involved in defining and evaluating software engineering approaches without considering how (or whether) developers will use and benefit from them. As a specific example, the talks presents a human study that shows how a family of techniques that received a great deal of attention in the last decade seemed to be ineffective when evaluated on real users.

## 5.17 My flings with data analysis

*Venkatesh-Prasad Ranganath (Kansas State University, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Venkatesh-Prasad Ranganath

**Joint work of** Ranganath, Venkatesh-Prasad; Vallathol, Pradip; Gupta, Pankaj  
**Main reference** V.-P. Ranganath, P. Vallathol, P. Gupta, “Compatibility Testing via Patterns-Based Trace Comparison,” in Proc. of the 29th ACM/IEEE Int’l Conf. on Automated Software Engineering (ASE’14), pp. 469–478, ACM, 2014; pre-print available from author’s webpage.  
**URL** <http://dx.doi.org/10.1145/2642937.2642942>  
**URL** <http://research.microsoft.com/apps/pubs/?id=171616>

This talk presents observations and opportunities in the space of using data analysis to enable, accomplish, and improve software engineering tasks such as testing. The observations and opportunities are drawn from efforts in an industrial setting.

## 5.18 Towards the Impact of Software Analytics

*Guenther Ruhe (University of Calgary, CA)*

**License** © Creative Commons BY 3.0 Unported license  
© Guenther Ruhe

**Joint work of** Ruhe, Guenther; Nayebi, Maleknaz; S M D. Al Alam  
**Main reference** M. Nayebi, G. Ruhe, “Analytical Product Release Planning,” to appear in C. Bird, T. Menzies, T. Zimmermann, eds., “The Art and Science of Analyzing Software Data”, Morgan-Kaufmann.

This position statement raises some questions related to the impact of software data analytics. Questions being raised are:

- How much of the analytics results actually have been used?
- Which actual decisions have been supported?
- How much of the results was useful?
- How much data analytics is enough?

Two examples of ongoing research are presented:

1. Software release readiness
2. Analytical product release planning

## 5.19 Mere Numbers aren’t Enough – Focus on Interpretation and Visualization

*Per Runeson (Lund University, SE)*

**License** © Creative Commons BY 3.0 Unported license  
© Per Runeson

Software analytics involve data collection, analysis, interpretation and visualization. Current research focus to a vast majority on the data and analysis. However, interpretation and visualization are more related to the use and utility of the software analytics. Through a few examples [2, 3], I show how the interpretation and visualization parts play a significant role, and I encourage to take these aspects into account in future research [1].

This talk is based on the following publications:

## References

- 1 Hassan, A. E., A. Hindle, P. Runeson, M. Shepperd, P. Devanbu, and S. Kim (2013). *Roundtable: What's Next in Software Analytics*. IEEE Software 30(4), 53–56.
- 2 Borg, M., P. Runeson, and A. Ardö(2013). *Recovering from a Decade: A Systematic Map of Information Retrieval Approaches to Software Traceability*. Empirical Software Engineering. DOI: 10.1109/MS.2006.147.
- 3 Engström, E., M. Mäntylä, P. Runeson, and M. Borg (2014). *Supporting Regression Test Scoping with Visual Analytics*. In: Proceedings of the 2014 IEEE International Conference on Software Testing, Verification, and Validation, pp.283-292. DOI: 10.1109/ICST.2014.41.

## 5.20 Composable Data Mining: Supporting Analytics for End Users

Anita Sarma (*University of Nebraska – Lincoln, US*)

License © Creative Commons BY 3.0 Unported license  
© Anita Sarma

Joint work of Sarma, Anita; Jose Ricardo da Silva, Leonardo Murta

There is a need for supporting end users in performing analytics of their own data. We provide a framework that translates the relationship among project elements into matrices and allows linear transformation of these matrices to combine relationships and provide insight. This talk shows how Dominoes can be used to identify expertise for a given project or an artifact (file) by considering not only the number of edits that a developer has made, but also the spread of their changes and thereby the breadth of their expertise. Our approach enables us to identify expertise over any given granularity and time period quickly.

## 5.21 42 years of Unix history in one repository

Diomidis Spinellis (*Athens University of Economics and Business, GR*)

License © Creative Commons BY 3.0 Unported license  
© Diomidis Spinellis

The goal of the Unix history repository project is to create a git repository representing the Unix source code history, starting from the early 1970s and ending in the modern time. To fulfil this goal the project brings data from early system snapshots, repositories, and primary research. The project aims to put in the repository as much meta-data as possible, allowing the automated analysis of Unix history. This effort allows the exploration of programming style evolution, the consolidation of digital artefacts of historical importance, the collection and recording of history that is fading away, and the provision of a data set for digital archaeology and repository mining. The project has achieved its first major goal with the establishment of a continuous time-line from 1972 to 2014. The repository contains snapshots of V1, V3, V4, V5, V6, and V7 Research Edition, Unix/32V, all available BSD releases, the CSRG SCCS history, two releases of 386BSD, FreeBSD 1.0, and an import of the FreeBSD repository starting from its initial imports that led to FreeBSD 2.0. The files appear to be added in the repository in chronological order according to their modification (or commit) time, and large parts of the source code have been attributed to their actual authors. Commands such as git blame and (sometimes) git log produce the expected results. The community can contribute to the project by using it for research, adding material, and proposing corrections. The repository is available online at <https://github.com/dspinellis/unix-history-repo>.



## 5.22 Open Problems and Challenges in Software Analytics

*Diomidis Spinellis (Athens University of Economics and Business, GR)*

License  Creative Commons BY 3.0 Unported license  
© Diomidis Spinellis


We identify open problems and challenges in software analytics in the areas of the domains that can be addressed, data analysis, and under-represented stakeholders. In terms of domains, we refer to the recent paper by Begel and Zimmermann on 145 questions for data scientists in software engineering, and outline a procedure that can be used to map these questions into domain challenges. In terms of data analysis, we identify the problems and challenges of linking persons with their actions in repositories, issue databases, mailing lists, and social networking sites, linking artefacts between systems that hold them (e.g. commits with issues they resolve), scalability of tools and techniques over large data sets, the sharing of industrial data, data privacy, the cleaning of noise, judging and dealing with data quality, judging the representativeness of data, and reproducing results. In terms of stakeholders, we highlight that build engineers, system administrators, people with multiple tasks in a software project, software designers/architects, and business/product managers, and support personnel are not well catered by current data analysis techniques.

### References

- 1 Andrew Begel, Thomas Zimmermann. Analyze This! 145 Questions for Data Scientists in Software Engineering. In Proceedings of the *36th International Conference on Software Engineering (ICSE 2014)*, Hyderabad, India, June 2014.

## 5.23 Studying social media in software development: reflecting on research methods

*Margaret-Anne Storey (University of Victoria, CA)*

License  Creative Commons BY 3.0 Unported license  
© Margaret-Anne Storey

I present a brief overview of research on the impact of social media on software engineering. This research highlights how software engineers today actively benefit from a participatory culture of development, that is fuelled by socially enabled tools. I also provide a brief overview of the landscape of research methods, highlighting some methods in particular: grounded theory and mixed methods. As I describe these methods, I reflect on my experiences using those methods to investigate and learn about social media use in software engineering. Finally, I suggest that we try to adopt some of the ways open source developers benefit from their participatory culture, so that we can improve and accelerate the research we do. By collaborating more, we are more likely to be able to use multiple methods to investigate software development which will help balance the different limitations of methods.

## 5.24 The Graph

*Burak Turhan (University of Oulu, FI)*

**License** © Creative Commons BY 3.0 Unported license  
© Burak Turhan

**Joint work of** Turhan, Burak; Taipale, Taneli; Qvist, Mika; Kuutti, Kari

**Main reference** T. Taipale, B. Turhan, M. Qvist, “Constructing Defect Predictors and Communicating the Outcomes to Practitioners”, in Proc. of the 7th Int’l Symp. on Empirical Software Engineering and Measurement (ESEM’13, Industry Track), pp. 357–362, IEEE, 2013.

**URL** <http://dx.doi.org/10.1109/ESEM.2013.45>

In this talk I’ll share our experiences in conducting a software analytics project, e. g. bug prediction. Though the project was a huge success in terms of scholarly outcomes and performance measures, we had difficulty in communicating the results to the practitioners. It turned out that our predictions were perceived as stating the obvious – even through many different modes of communication/ representation; and the most useful and insightful representation was only a visualization of the issue reports without any predictions. Hence, exploration trumps prediction in our case!

## 5.25 Why Quality Models Don’t Work and Why We Need Them Anyway

*Stefan Wagner (Universität Stuttgart, DE)*

**License** © Creative Commons BY 3.0 Unported license  
© Stefan Wagner

**Joint work of** Wagner, Stefan; Lochmann, Klaus; Heinemann, Lars; Kläs, Michael; Trendowicz, Adam; Plösch, Reinhold; Seidl, Andreas; Goeb, Andreas; Streit, Jonathan

**Main reference** S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidl, A. Goeb, J. Streit, “The Quamoco Product Quality Modelling and Assessment Approach,” in Proc. of 34th Int’l Conf. on Software Engineering (ICSE’12), pp. 1133–1142, IEEE, 2012

**URL** <http://dx.doi.org/10.1109/ICSE.2012.6227106>

Quality is a complex and multifaceted concept. We employ quality models to capture this complexity. In the project Quamoco, we built a detailed and operationalised quality model connecting low-level metrics with high-level quality attributes. We managed to build a model judged well understandable by developers and giving reasonable quality estimates. Yet, the effort that went into it is prohibitive for a real general model. Also, the many factors in the model makes it almost impossible to really validate it. So where should we go? Staying with predicting other low-level metrics (such as defects) or put in the effort to describe the relationships to high-level attributes?

## 5.26 Analytics in the Field

*Patrick Wagstrom (IBM Watson Group, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Patrick Wagstrom

As researchers we often develop elegant tools and models that explain phenomena behind software engineering. These models are purported to make the development process faster, easier, and more reliable. However, these research outputs are rarely taken up in industry. This talk, presented from the industry perspective, identifies some of the challenges around deploying research output in industry from the lens of understanding user interactions and desires.

## 5.27 Software Analytics in Practice

*Dongmei Zhang (Microsoft Research – Beijing, CN)*

**License** © Creative Commons BY 3.0 Unported license  
© Dongmei Zhang

**Joint work of** Zhang, Dongmei; Shi Han, Yingnong Dang, Jian-Guang Lou, Haidong Zhang, Tao Xie  
**Main reference** D. Zhang, S. Han, Y. Dang, J.-G. Lou, H. Zhang, T. Xie, “Software Analytics in Practice,” IEEE Software – Special Issue on the Many Faces of Software Analytics, 30(5):30–37, 2013; pre-print available from author’s webpage.

**URL** <http://dx.doi.org/10.1109/MS.2013.94>

**URL** <http://research.microsoft.com/en-us/groups/sa/ieeesoft13-softanalytics.pdf>

In this short talk, I’ll introduce our definition of software analytics, and explain the definition from five perspectives – research topics, target audience, input/output, technology pillars, and connection to practice. In particular, I’ll discuss the importance of connection to practice, because (1) the data under study comes from real practice; (2) there are real problems to be answered using the data; (3) one of the success metrics of software analytics research is its influence and impact on the development practice.

## 6 Breakout Groups

We will now briefly describe six breakout sessions that were held. The materials produced by the groups are archived at <http://www.dagstuhl.de/mat/index.en.phtml?14261>.

- **Sharing data, methods, and models.** The breakout did a SWOT (Strength, Opportunity, Weakness and Threats) analysis of data, method and model sharing. In general, all sharing was deemed beneficial for sharing the workload, generalizability, validation, and collaboration. However, putting the shared artifacts in context was a threat in all cases.
- **Industry collaboration.** The breakout focused on issues related to this important collaboration. It was recognized that each of industry and research have different needs but working together is essential and beneficial. For example, research desires statistical significance while this is not a concern to industry. Trust must be established between both parties. Potential legal issues to enable the collaboration and data sharing may arise.
- **Development of a software analytics development community.** The breakout group formed a GitHub organization to maintain lists of Software Development Analytics Community blogs and other resources.
- **Tools and techniques.** Tools for quantitative and qualitative methods were discussed. It was suggested to create a wiki for the classification of methods, questions asked, good examples of case studies, pitfalls.
- **Analytics Framework.** The group discussed a spreadsheet to identify dimensions of analytics, to identify questions asked for each cell in the n-dimensional framework and to link papers that address those questions
- **Manifesto.** Significant time was spent to develop a data analytics manifesto, as will be discussed in the next section.

Another outcome of a breakout session were “Good Practices for Software Analytics Papers”, which can be found at this URL: <http://www.cs.mcgill.ca/~martin/blog/2014-06-26.html>

## Participants

- Bram Adams  
Polytechnique Montreal, CA
- Alberto Bacchelli  
TU Delft, NL
- Ayse Bener  
Ryerson Univ. – Toronto, CA
- Trevor Carnahan  
Microsoft Res. – Redmond, US
- Serge Demeyer  
University of Antwerp, BE
- Premkumar T. Devanbu  
Univ. of California – Davis, US
- Stephan Diehl  
Universität Trier, DE
- Michael W. Godfrey  
University of Waterloo, CA
- Alessandra Gorla  
Universität des Saarlandes –  
Saarbrücken, DE
- Georgios Gousios  
TU Delft, NL
- Mark Grechanik  
Univ. of Illinois – Chicago, US
- Michaela Greiler  
Microsoft Res. – Redmond, US
- Abram Hindle  
University of Alberta, CA
- Reid Holmes  
University of Waterloo, CA
- Miryung Kim  
University of Texas – Austin, US
- Andrew J. Ko  
University of Washington –  
Seattle, US
- Lucas M. Layman  
Fraunhofer USA, US
- Andrian Marcus  
Wayne State University, US
- Nenad Medvidovic  
Univ. of Southern California, US
- Tim Menzies  
West Virginia University –  
Morgantown, US
- Leandro L. Minku  
University of Birmingham, GB
- Audris Mockus  
Avaya – Basking Ridge, US
- Brendan Murphy  
Microsoft Res. – Cambridge, GB
- Meiyappan Nagappan  
Rochester Institute of  
Technology, US
- Alessandro Orso  
Georgia Inst. of Technology, US
- Martin Pinzger  
Universität Klagenfurt, AT
- Denys Poshyvanyk  
College of William and Mary –  
Williamsburg, US
- Venkatesh-Prasad Ranganath  
Kansas State University, US
- Romain Robbes  
University of Chile, CL
- Martin Robillard  
McGill University, CA
- Guenther Ruhe  
University of Calgary, CA
- Per Runeson  
Lund University, SE
- Anita Sarma  
Univ. of Nebraska – Lincoln, US
- Emad Shihab  
Concordia Univ. – Montreal, CA
- Diomidis Spinellis  
Athens University of Economics  
and Business, GR
- Margaret-Anne Storey  
University of Victoria, CA
- Burak Turhan  
University of Oulu, FI
- Stefan Wagner  
Universität Stuttgart, DE
- Patrick Wagstrom  
IBM Watson Group, US
- Jim Whitehead  
University of California – Santa  
Cruz, US
- Laurie Williams  
North Carolina State Univ., US
- Dongmei Zhang  
Microsoft Res. – Asia, CN
- Thomas Zimmermann  
Microsoft Res. – Redmond, US

