Analytics for Software Development

Raymond P.L. Buse*
The University of Virginia
buse@cs.virginia.edu

Thomas Zimmermann
Microsoft Research
tzimmer@microsoft.com

ABSTRACT

Despite large volumes of data and many types of metrics, software projects continue to be difficult to predict and risky to conduct. In this paper we propose *software analytics* which holds out the promise of helping the managers of software projects turn their plentiful information resources, produced readily by current tools, into insights they can act on. We discuss how analytics works, why it's a good fit for software engineering, and the research problems that must be overcome in order to realize its promise.

Categories and Subject Descriptors

D.2.9 [**Management**]

General Terms

management, human factors, measurement

Keywords

analytics, project management

1. INTRODUCTION

Software engineering is an exceedingly data rich activity. Nearly every artifact of a project's development, from code repositories to testing frameworks to bug databases, can be measured with a high degree of automation, efficiency, and granularity. Projects can be measured throughout their life-cycle: from specification to maintenance. The research community has proposed numerous metrics and models for complexity, maintainability, readability, failure propensity and many other important aspects of software quality and development process health (e.g., [7, 16]).

Nonetheless, software development continues to be risky and unpredictable. It is not unusual for major development efforts to experience large delays or failures [1]. Moreover, software defects cost the US economy many billions of dollars each year [19].

Together, these observations imply that there continues to be a substantial disconnect between (A) the information needed by project managers to make good decisions and (B) the information currently delivered by existing tools. At its root, the problem is that the real-world information needs of project managers are not well understood by the research community. Indeed, research has largely ignored the needs of managers and has instead focused on the information needs of developers (e.g., [3, 14, 20]).

When data needs are not met, either because tools are unavailable, too difficult to use, too difficult to interpret, or they simply do not present useful or actionable information, managers must primarily rely on past experience and intuition for critical decision making. Such intuition-based decisions can sometimes work out well, but often they are unnecessarily suboptimal or even destructive [23]. As software projects continue to grow in size and complexity, decision making will likely only become more difficult.

The information-intensive nature of software engineering coupled with other features which we will discuss later suggests that a strong potential exists for software project management to make great use of analysis, data, and systematic reasoning to make decisions. This data-centric style of decision making is known as *analytics*. The idea of analytics is to leverage potentially large amounts of data into real and actionable insights.

The concerted use of analytics has revolutionized decision making across many fields [10]. Many in the technology community are probably familiar with web analytics [13]. Web analytics leverages large volumes of click-stream data to help website managers make informed decisions about many aspects of their business from advertising to content layout to investment. Today, large websites not only thoroughly test all proposed changes in the traditional sense, but they also undertake detailed analytic experiments aimed to precisely quantify the net benefit of any proposed change. Analytics has had a profound effect on businesses ranging from technology to retail to financial.

We believe that analytics techniques hold great promise for software development. Software, however, presents a number of new challenges related both to the complexity of the code itself, but also to the complexity of project management. In this paper we discuss a number of important research challenges that must be met before analytics can be successfully applied in this context.

First, we believe that software project management is

*Raymond Buse was an intern at Microsoft Research when this paper was written.

Microsoft Research. Technical Report MSR-TR-2010-111.
© 2010 Microsoft Corporation. All rights reserved.

poorly understood. In Section 2, we advocate new in-depth *studies of the information needs* of those who make critical decisions in software projects. In order to assist managers, it is critically important to first understand how they work: What conditions do they look for? What actions can they take?

Second, we need new and principled *tools for data aggregation and analysis* suitable for use by project managers. In Section 4 we discuss the limitations of current tools and some of the possibilities for future ones.

Finally, in Section 5 we advocate changes to the development management process itself. In particular, we propose the position of *software project analyst*. Such a position shares both the analytic skills and domain knowledge necessary to execute powerful analyses beyond the scope of both mangers and tools.

We begin by discussing the current understanding of project management and why it remains insufficient to enable the successful application of analytics to software development.

2. PROJECT MANAGEMENT

Software project management is a complex and broadly defined position. Project managers monitor and guide the work of some number of designers, developers and testers of software while sometimes participating in these activities themselves. Where engineers focus on code, architecture, and performance, managers focus on high level concerns: the direction of the project, allocation of resources and in some cases details of the user experience, the feature set, the way the product will get used. Managers work to simultaneously satisfy (potentially conflicting) constraints imposted by customers, developers, testers, maintainers, and management. Steven Sinofsky, a manager at Microsoft, notes that "it is almost impossible to document a complete list of the responsibilities of program managers" [22].

The complexity of the job of a project manager contributes to the difficulty of designing and evaluating tools by the research community. In particular, the information needs of managers are not well understood. Boehm and Ross proposed a theory of project management which included the "top 10 primary sources of software project risk and the most effective approaches for resolving them" [6]. While top ten lists like this can be instructive, the problem is that the management techniques presented (e.g., benchmarking, organization analysis, technical analysis, etc.) aren't specific enough to be useful. Many critical questions remain unanswered: Which of these can be performed automatically? Which are most important? How should the results be presented? What decisions can they lead to? How does one evaluate success?

More recently, in an effort to begin answering some of these questions, Wallace *et al.* conducted a survey of 507 project managers [26]. Cluster analysis was used in an attempt to identify risk factors in projects. That study found, for example, that "even low risk projects have a high level of complexity." The study did not produce a practical answer to the question of what software indicators managers should be concerned with.

Komi-Sirvio *et al.* noted that software managers are typically too busy with their day-to-day duties to spend much time performing measurement activities [15]. Typically data-driven tasks are relegated to secondary work. We believe that this should be changed. However, much more study is needed for the research community is to understand project managers well enough to pursue new process models and tools for them in a principled way. In the next section we describe how the application of analytics represents a means of bridging the gap between managers and the data they need to make informed decisions.

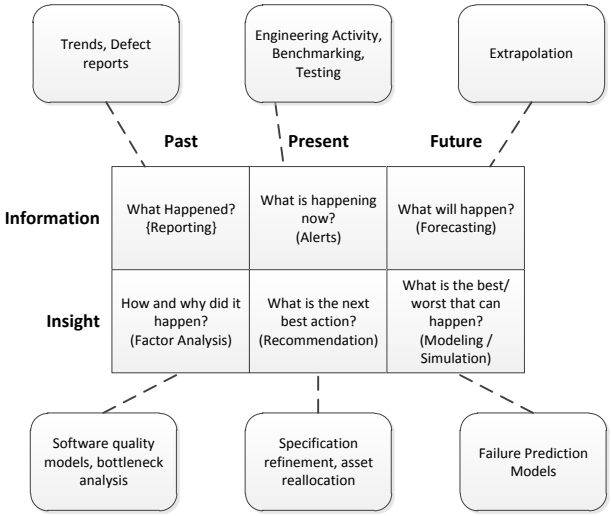

Figure 1: Analytical Questions (adapted from [10]): We distinguish between questions of *information* which can be directly measured, from questions of *insight* which arise from a careful analytic analysis and provide managers with a basis for action.

3. SOFTWARE ANALYTICS

Analytics can help answer important questions mangers ask about their projects. The goal of analytics to assist decision makers in extracting important information and insights from data sets that would otherwise be hidden.

Figure 1, which we adapt from Davenport *et al.* [10], identifies six question areas analytics can help answer organized by time-frame and by information vs. insight. The idea is to distinguish questions of *information* which some tools already provide (e.g., how many bugs are in the bug database?) from questions of *insight* which provide managers with an understanding of a project's dynamics and a basis on which to make decisions (e.g., will the project be delayed?).

The overarching goal of analytics is to help managers move beyond information and toward insight. However, such a transition isn't easy. Insight necessarily requires knowledge of the domain coupled with the ability to identify patterns involving multiple indicators. For example, to understand why the number of bug reports is increasing and if it is a concern, it's important to understand what other activities are occurring: Is the next release soon? Is the team working on bug fixes or feature additions? Analytical techniques can help managers quickly find important needles in massive haystacks of data.

The rounded boxes in Figure 1 provide examples of how analytics can fit well with traditional software engineering metrics and concepts. In fact, software engineering has many qualities that suggest a business process that lends

itself well to analytics:

- **Data-rich.** Analytics operates best when large amounts of is data available for analysis.
- **Labor intensive.** Analytics enable leverage of expertise especially where talent supply is short, demand is cyclical, and training times are lengthy [10]. Software engineering is especially labor intensive. Furthermore, numerous studies have found an order of magnitude productivity gap between developers including [5, 25].
- **Timing dependent.** Analytics can be helpful in cases where business products must meet specific schedules; analytics enable decision makers to look both upstream and downstream.
- **Dependent on consistency and control.** Analytics help enable consistent decision making even under unusual circumstances [10].
- **Dependent on distributed decision making.** Analytics can help decision makers understand the collective state of projects even in the face of great geographic distribution. Many software projects feature highly distributed development, especially in open source projects.
- **Low average success rate.** Domains with a high failure rate are the most likely to benefit from analytics. Software projects fail as much as 33% of the time [1].

There are many potential advantages to application of rigorous analytics use to software project management (inspired by [10]). Analytics can help mangers:

- **Monitor a project.** Analytics provides tools to help managers understand the dynamics of a complex project.
- **Know what's really working.** Analytics can help evaluate the effectiveness of a change to the development process. For example, it can measure whether some effect is statistically significant.
- **Improve efficiency.** Techniques based on analytics can help mangers allocate resources and talent to where it is most needed and recognize when under utilization occurs.
- **Manage risk.** The effectiveness of risk models can be improved with increases in the quantity and precision of data. Analytics can provide both a data channel into risk models and a vehicle for delivering the output of such models in an intuitive way.
- **Anticipate changes.** Analytics can help managers to detect and forecast trends in data.
- **Evaluate past decisions.** Logical and consistent decision making based on data is much more amenable to later review and assessment than decisions based on intuition.

Analytics helps describe a reasoning framework which we've observed has the potential to fit well with software engineering. However, to realize that potential it is obvious that beyond studies, tool support is also necessary. In the next section we discuss some of the tools that are currently available and why they are not yet sufficient for this task.

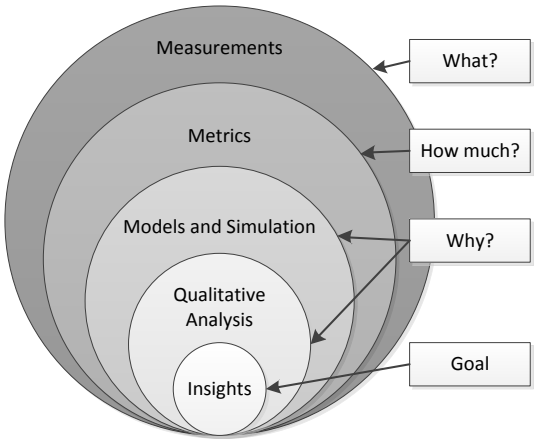

Figure 2: Paradigm of Analytics (adapted from [13]): The role of the analyst is to compose many types analyses to formulate more complete insights.

4. ANALYTICS TOOLS

There are a number of existing tools designed to support management. For example, `PROM` [21] and `Hackystat` [12] are both capable of monitoring and reporting a great number of software project statistics. However, after seven and ten years of development respectively, neither have seen significant levels of adoption outside of academia [11]. One explanation might be that these tools simply do not answer the right questions [8]. Each of these tools focus primarily on data collection, which while a challenging problem in of itself is probably no longer the most critical concern.

More recent tools, which have focused on data presentation rather than collection have met with some adoption. Microsoft's `Team Foundation Server` and IBM's `Jazz` developer environment [9], for example, provide *dashboard* views designed to keep developers up-to-date on the status of various events like modifications, bugs, and build results. However, a recent study concluded that while integrated tooling of this type could help support the development process, "the distinction between high-level and low-level awareness is often unclear" [24].

While modern tools can present a large amount of data from varied sources, most either focus on data collection or on developer *awareness*; because they don't have a good model for the needs of real product managers, real product managers do not generally use them.

Moreover, managers may be too busy or may simply lack the quantitative skills or analytic expertise to fully leverage advanced analytical applications which may range from trend analysis, classification algorithms, predictive modeling, statistical modeling, optimization and simulation, and data- and text-mining [10]. One possibility is that tools should be created with this in mind: that they are to be consumed by managers with little expertise and great time constraints. However, another intriguing possibility is the addition of an analytic professional to the software development team.

5. SOFTWARE ANALYSTS

An *analyst* is an expert in data creation, collection, interpretation and use, but is also trained in the workings of the business process in question. Consider Figure 2 which we adapt from Kaushik [13]. In current practice, managers sometimes glean sparse insights purely from measurements or metrics. Other managers may rely primarily on qualitative discussions. The role of the analyst is to use quantitative skill and domain knowledge to combine many types of quantitative and qualitative information and form the most complete insights.

Software analysts could be enlisted to perform studies too involved for managers or even for sophisticated tools. For example, Bird *et al.* found that distributed development did not appreciably effect software quality in the implementation of Windows Vista [4]. Another study by Mockus *et al.* enumerated a set of metrics which may be highly predictive of defects [17]. An analyst could carry out similar studies and prescribe corrective action based on the results. Furthermore, many findings in software engineering research depend on large numbers of context variables [2]. As such, these findings may not generalize [18, 27]. A software analyst is critical for determining which important results offered by the research community apply to a specific project.

We believe that analysts could be trained as part of a Software Engineering curriculum modified to emphasize quantitative skills, or through a Master of Analytics program like the one launched at North Carolina State University.[1] Research into software analytics as advocated by this paper could constitute an invaluable guide for study in this area.

6. CONCLUSION

All resources, especially talent, are always constrained. This alludes to the importance of careful and deliberate decision making by the managers of software projects. The observation that software projects continue to be risky and unpredictable despite being highly measurable implies that more analytic information should be leveraged toward decision making. In this paper, we described how software analytics can help managers move from low-level measurements to high-level insights about complex projects. We advocated more research into the information needs and decision process of managers in order that we might build new tools capable of meeting those needs and revealing information on which managers can make better decisions. Finally, we discussed how the complexity of software development suggests that dedicated analytic professionals with both quantitative skills and domain knowledge might provide great benefit to future projects.

7. REFERENCES

[1] T. Addison and S. Vallabh. Controlling software project risks: an empirical study of methods used by experienced project managers. In *SAICSIT '02*, pages 128–140, 2002.

[2] V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25:456–473, 1999.

[3] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson. Fastdash: a visual dashboard for fostering awareness in software teams. In *CHI '07*, pages 1313–1322, 2007.

[4] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does distributed development affect software quality?: an empirical case study of windows vista. *Commun. ACM*, 52(8):85–93, 2009.

[5] B. Boehm. *Software Cost Estimation with Cocomo II.* Addison Wesley, Boston, MA, 2000.

[6] B. Boehm and R. Ross. Theory-w software project management principles and examples. *IEEE TSE*, 15(7):902 –916, jul 1989.

[7] R. P. L. Buse and W. R. Weimer. A metric for software readability. In *International Symposium on Software Testing and Analysis*, pages 121–130, 2008.

[8] I. D. Coman, A. Sillitti, and G. Succi. A case-study on using an automated in-process software engineering measurement and analysis system in an industrial environment. In *ICSE '09*, pages 89–99, 2009.

[9] I. Corporation. Jazz. `http://www.ibm.com/software/rational/jazz/`.

[10] T. Davenport, J. Harris, and R. Morison. *Analytics at Work.* Harvard Business School Publishing Corporation, Boston, MA, 2010.

[11] G. Gousios and D. Spinellis. Alitheia core: An extensible software quality monitoring platform. In *ICSE '09*, pages 579–582, 2009.

[12] P. Johnson, H. Kou, M. Paulding, Q. Zhang, A. Kagawa, and T. Yamashita. Improving software development management through software project telemetry. *Software, IEEE*, 22(4):76 – 85, july-aug. 2005.

[13] A. Kaushik. *Web Analytics 2.0.* Wiley Publishing, 2010.

[14] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. *Software Engineering, International Conference on*, 0:344–353, 2007.

[15] S. Komi-Sirvi, P. Parviainen, and J. Ronkainen. Measurement automation: Methodological background and practical solutions-a multiple case study. *Software Metrics, IEEE International Symposium on*, 0:306, 2001.

[16] T. J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.

[17] A. Mockus, P. Zhang, and P. L. Li. Predictors of customer perceived software quality. In *ICSE '05*, pages 225–233, 2005.

[18] I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Trans. Softw. Eng.*, 31(5):380–391, 2005.

[19] National Institute of Standards and Technology. The economic impacts of inadequate infrastructure for software testing. Technical Report 02-3, Research Triangle Institute, May 2002.

[20] J. Sillito, G. C. Murphy, and K. De Volder. Questions programmers ask during software evolution tasks. In *SIGSOFT '06/FSE-14*, pages 23–34, 2006.

[21] A. Sillitti, A. Janes, G. Succi, and T. Vernazza. Collecting, integrating and analyzing software metrics and personal software process data. In *EUROMICRO*, page 336, 2003.

[22] S. Sinofsky. Steven sinofsky's microsoft techtalk / pm at microsoft. `http://blogs.msdn.com/b/techtalk/archive/2005/12/16/504872.aspx`, Dec. 2005.

[23] L. Strigini. Limiting the dangers of intuitive decision making. *IEEE Software*, 13:101–103, 1996.

[24] C. Treude and M.-A. Storey. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In *ICSE '10*, pages 365–374, 2010.

[25] J. D. Valett and F. E. McGarry. A summary of software measurement experiences in the software engineering laboratory. *Journal of Systems and Software*, 9(2):137 – 148, 1989.

[26] L. Wallace, M. Keil, and A. Rai. Understanding software project risk: a cluster analysis. *Inf. Manage.*, 42(1):115–125, 2004.

[27] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction. In *Symposium on the Foundations of Software Engineering*, August 2009.

[1] `http://analytics.ncsu.edu`