Poster: Precooked Developer Dashboards: What to Show and How to Use

Vladimir Ivanov
Innopolis University
Universitetskaya St, 1
Innopolis, Respublika Tatarstan,
Russia 420500
v.ivanov@innopolis.ru

Alan Rogers
Innopolis University
Universitetskaya St, 1
Innopolis, Respublika Tatarstan,
Russia 420500
a.rogers@innopolis.ru

Giancarlo Succi
Innopolis University
Universitetskaya St, 1
Innopolis, Respublika Tatarstan,
Russia 420500
g.succi@innopolis.ru

Jooyong Yi
Innopolis University
Universitetskaya St, 1
Innopolis, Respublika Tatarstan,
Russia 420500
j.yi@innopolis.ru

Vasili Zorin
Innopolis University
Universitetskaya St, 1
Innopolis, Respublika Tatarstan,
Russia 420500
v.zorin@innopolis.ru

ABSTRACT

Designing an effective and useful dashboard is expensive and it would be important to determine if it is possible to elaborate a "generic" useful and effective dashboard, usable in a variety of circumstances. To determine if it is possible to develop such dashboard and, if so, its structure we interviewed 67 software engineers from 44 different companies. Their answers made us confident in the possibility of building such dashboard.

ACM Reference format:
Vladimir Ivanov, Alan Rogers, Giancarlo Succi, Jooyong Yi, and Vasili Zorin. 2018. Poster: Precooked Developer Dashboards:
What to Show and How to Use. In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
DOI: 10.1145/3183440.3195028

1 INTRODUCTION

There is increasing interest and use of developer dashboards in the software industry [2, 4, 10, 13, 14]. Developer dashboards are typically used to visualize the overall status of a project — the assumption here is that visualization helps managers/developers be aware of the overall status of the projects they are working on, and make proper collaborative/individual decisions while developing software, which will improve the overall productivity and reliability of a team [4–6, 8, 14, 15] and avoiding misinterpretation of data [3, 9]. However, this promising assumption hold only if a dashboard displays information needed by the users, without undesired and distracting details.

What kinds of information do developers want to see in a dashboard? And, why do developers want to see them in a dashboard? Also, how do developer want to use dashboards in their development activities? These are the key research questions we ask in this study. While there would be no single answer to these questions [7], we seek to find general answers from software engineers in the field. Despite that most modern dashboards are customizable, developers often use the default dashboard due to the cost entailed by customization — for example, developers often do not want to read through the full functionalities the dashboard provides, and consequently do not customize the dashboard, as reported in [14].

To answer our research questions, we conduct face-to-face individual interviews with 67 software engineers from at least 44 different companies (some developers did not want the name of their organization to be revealed). We designed our survey questionnaire/sessions, taking into account the Goal-Question-Metric (GQM) approach [1]. Knowing how developers want to use dashboards is essential when designing an effective dashboard.

Our key contributions are: *(1) identifying metrics developers want to see in dashboards with an observational study [12], (2) providing a GQM model to understand the relevance of the identified metrics, and (3) determining how developers want to use dashboards.*

2 THE SURVEY

To approach our problem, we run a survey, collecting information from developers to build a "typical" GQM, from which to derive the dashboard. The following research **questions** were identified:

1. What are the most critical issues that software development companies face during the production process

2. What information do companies use to detect problems during the software development process?

3. Which metrics do companies use to detect problems during the software development process?

4. Which structures and functions are appropriate for a dashboard?

A dashboard typically visualizes quantified information, and thus answering our first research question essentially boils down to figuring out metrics developers want to see in a dashboard. We use a goal-directed GQM approach to find metrics developers want to see in a dashboard. Through identifying typical goals developers want to achieve, typical questions developers ask to pursue the goals,

ICSE '18 Companion, Gothenburg, Sweden
 978-1-4503-5663-3/18/05…$15.00
DOI: 10.1145/3183440.3195028

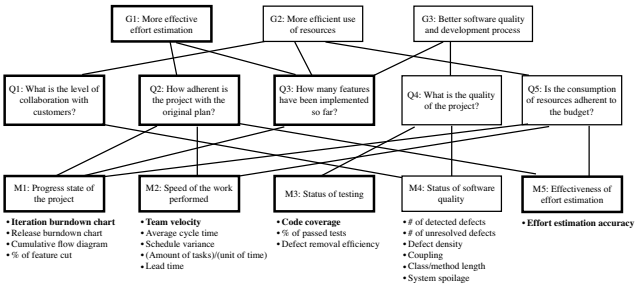

Figure 1: The resulting "typical" GQM model where strong labels and high-frequency metrics are depicted in boldface.

and finally typical metrics that can answer the typical developer questions, we obtain a GQM model (Figure 1). The obtained GQM model shows how the identified typical metrics are related to the identified typical goals, and thus can be used to understand the reasons developers want to see the identified metrics in a dashboard.

3 GOALS, QUESTIONS AND METRICS FOR PRECOOKED DASHBOARDS

The resulting typical GQM model (Figure 1) shows a summary of the results of our survey —it encompasses all responses to our survey questions asked in three different abstraction levels. Our GQM model also shows, with bold-line boxes, which goals/questions/metrics our informants express particularly strong interest; we assign bold-line boxes to goals/questions/metrics labeled as strong. An immediate usage of our GQM model is for building a precooked (default) developer dashboard that is likely to be useful for typical developers. It is noteworthy that our GQM model can be applied to a wide range of companies to which our informants are employed, as we have shown previously.

4 HOW DO DEVELOPERS WANT TO USE A DASHBOARD?

Overall, we summarize our survey results as follows:

> According to our survey, developers are more interested in using dashboards for their operational needs such as monitoring performance than for strategic/analytic purposes. Also, more number of our subjects prefer the push mode to the pull mode, coherently with the fact that the most desired feature is alerting significant deviation from expected values.

In details, answering this cognitive question ("how to display/notify these contents in a dashboard?") thoroughly is beyond the capacity of our survey, and user experiments would also be required. Still, our survey provides an opportunity to answer a related more specific question, "how do developers want to use a dashboard?", and we show in the following the results for this last part of our survey.

An operational dashboard (where detailed information about software development process is constantly monitored) is preferred over an analytical dashboard (where the causes of problems are analyzed) and a strategic dashboard (where a snapshot of the project is displayed). We measured the degree of the user preference for each type of a dashboard with a sequence score calculated using the gamma analysis [11], which places the user preference of each type of a dashboard between -1 (lowest preference) and 1 (highest).

5 CONCLUSIONS

In this study, we have conducted a survey with developers from various companies, in an attempt to obtain information necessary to build effective developer dashboards. We have identified the five distinct kinds of metrics: (1) metrics to show the current progress state of the project, (2) metrics to show the speed of work performed, (3) metrics to show the status of testing, (4) metrics to show the status of software quality, and (5) metrics to show the effectiveness of effort estimation. We have also connected these metrics with the goals developers want to achieve, through a GQM model. We have observed that, in general, developers are more concerned about monitoring whether the development process is on track than monitoring software quality.

We have spotted the difficulty of supporting this notification effectively — it is difficult to picture the right track, when developers are currently having difficulties in estimating the effort necessary to complete a task. For this reason, we argue that research on effort prediction deserves strong attention.

REFERENCES

[1] Victor R. Basili and David M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE Trans. Software Eng.* 10, 6 (1984), 728–738.
[2] Olga Baysal, Reid Holmes, and Michael W. Godfrey. 2013. Developer Dashboards: The Need for Qualitative Analytics. *IEEE Software* 30, 4 (2013), 46–52.
[3] Luigi Benedicenti, Paolo Ciancarini, Franco Cotugno, Angelo Messina, Alberto Sillitti, and Giancarlo Succi. 2017. Improved Agile: A Customized Scrum Process for Project Management in Defense and Security. In *Software Project Management for Distributed Computing*. Springer International Publishing, 289–314.
[4] Jan Bosch and Helena Olsson. 2017. Towards Evidence-Based Organizations: Learnings From Embedded Systems, Online Games And Internet of Things. *IEEE Software* PP, 99 (2017). Early Access Article.
[5] Ilenia Fronza, Alberto Sillitti, and Giancarlo Succi. 2009. An Interpretation of the Results of the Analysis of Pair Programming During Novices Integration in a Team. In *Proceedings of ESEM 2009 (ESEM '09)*. IEEE Computer Society, 225–235.
[6] Vladimir Ivanov, Alexey Reznik, and Giancarlo Succi. 2018. Comparing the reliability of software systems: a case study on mobile operating systems. *Information Science* 423 (2018), 398–411.
[7] Vladimir Ivanov, Alan Rogers, Giancarlo Succi, Jooyong Yi, and Vasily Zorin. 2017. What Do Software Engineers Care About? Gaps Between Research And Practice. In *Proceedings of the 2017 ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2017)*.
[8] Andrea Janes and Giancarlo Succi. 2014. *Lean Software Development in Action*. Springer, Heidelberg, Germany.
[9] Andrea A. Janes and Giancarlo Succi. 2012. The Dark Side of Agile Software Development. In *Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2012)*. ACM, New York, NY, USA, 215–228.
[10] Witold Pedrycz, Barbara Russo, and Giancarlo Succi. 2012. Knowledge Transfer in System Modeling and Its Realization Through an Optimal Allocation of Information Granularity. *Appl. Soft Comput.* 12, 8 (Aug. 2012), 1985–1995. DOI: http://dx.doi.org/10.1016/j.asoc.2012.02.004
[11] Donald C Pelz. 1985. Innovation complexity and the sequence of innovating stages. *Science Communication* 6, 3 (1985), 261–291.
[12] Paul R. Rosenbaum. 2010. *Design of Observational Studies*. Springer, New York.
[13] Alberto Sillitti, Andrea Janes, Giancarlo Succi, and Tullio Vernazza. 2004. Measures for mobile users: an architecture. *Journal of Systems Architecture* 50, 7 (2004), 393–405.
[14] Christoph Treude and Margaret-Anne Storey. 2010. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In *ICSE*.
[15] Tullio Vernazza, Giampiero Granatella, Giancarlo Succi, Luigi Benedicenti, and Martin Mintchev. 2000. Defining Metrics for Software Components. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*.