Delft University of Technology
Software Engineering Research Group
Technical Report Series

Smart Views for Analyzing Problem Reports: Tool Demo

Patrick Knab, Harald C. Gall, and Martin Pinzger

Report TUD-SERG-2009-030

TUD-SERG-2009-030

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Note: Accepted for publication in the Proceedings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), 2009, ACM.

Smart Views for Analyzing Problem Reports: Tool Demo

Patrick Knab
Dep. of Informatics
Univ. of Zurich
Switzerland
knab@ifi.uzh.ch

Harald Gall
Dep. of Informatics
Univ. of Zurich
Switzerland
gall@ifi.uzh.ch

Martin Pinzger
Dep. of Software Technology
Delft Univ. of Tech.
The Netherlands
m.pinzger@tudelft.nl

ABSTRACT

Issue tracking repositories contain a wealth of information for reasoning about various aspects of software development processes. In this paper, we focus on bug triaging and provide visual means to explore the effort estimation quality and the bug life-cycle of reported problems.

Our approach uses a combination of graphical views to investigate details of individual problem reports while maintaining the context provided by the surrounding data population. This enables the detection and detailed analysis of hidden patterns and facilitates the analysis of problem report outliers.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Time estimation*

General Terms

Measurement

1. INTRODUCTION

In the EUREKA/ITEA[1] SERIOUS[2] project we worked together with industrial partners to *Investigate means to improve the planning and resource allocation in a software project by analysing change request and problem report data*.

We were given access to an issue tracking repository containing thousands of problem reports (PR) from a multi-year, multi-site software development project in the consumer electronics domain.

Since the statistical analysis of the data showed inconclusive results, mainly due to a strong presence of outliers and a heavily right-skewed data distribution, we developed a flexible interactive visualization approach which we present in this paper. The approach is based on the Micro/Macro Reading idea from Tufte and provides a combination of Overview and Details-on-demand. It currently comprises four simple and easy to understand views which can be linked in various ways. This integration enables sophisticated analyses by drilling down to details while preserving the big picture.

Other approaches for problem report visualization focus on very specific visualization techniques that are tailored to help answering well-defined but somewhat restricted questions. For example, D'Ambros *et al*. visualize the life-cycle of bugs [1] with the System Radiography and the Bug Watch View. They focus on time, activity and severity/priority aspects of bugs. Halverson *et al*. [2] focus on the social aspects and visualize state changes to reveal problematic patterns such as multiple resolve/reopen cycles.

Such approaches, while valuable for the specific task they were created for, are not easy to generalize, and depending on the amount of displayed information hard to understand. In contrast, our approach uses easy to understand visualizations which can be combined to provide a rich multidimensional view on the underlying problem report information. By using different view combinations and mappings (sort order, color, etc.) a wide range of analyses is possible and many aspects of the data can be explored. This flexility allows a user to follow his own exploration path depending on the current objectives.

We applied our interactive visualization technique to the problem report data provided by the industrial partner [3]. Using a combination of views, we found, for example, people who analyzed bugs with strong tendencies to either over- or underestimate the effort for solving PRs. We detected PRs that did not go through the analysis phase at the beginning of their life-cycle, which for several of them, led to complications (i.e., additional effort) later on. We further experienced that the relatively simple building blocks of our visualization facilitated communication with the project managers. Most participants of our meetings understood the visualizations in a matter of minutes and were therefore able to contribute valuable background information, as well as starting points for new examinations.

2. VISUALIZATION BUILDING BLOCKS

In this section we present the building blocks of our interactive visualization approach. These building blocks can be combined in a flexible and powerful way.

For the visualization of effort measures, we use **Polymetric Views** [4], the width of the boxes is determined by the value of the *estimatedEffort* and the height by the value of the *actualEffort*. The effort measure is the sum of all efforts that were exerted to resolve the issue described by the problem report. The color of the box is mapped to the identity of the analyzer (other mappings are also possible).

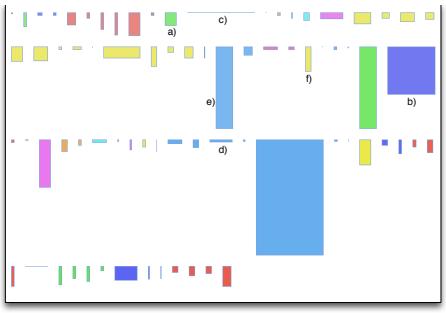
In Figure 4 we have a selection of PRs ordered by analyzer. With this ordering we can compare analyzers with regard to estimation accuracy, their tendency to over- or underestimate PRs, and we can also compare the efforts of different PRs. For example, the figure shows a few accurate estimators (*e.g.*, labels a and b), very few over-estimators (*e.g.*, labels c and d), and a large set of under-estimators (*e.g.*, labels e and f).

To visualize the duration of process steps we use a **pie chart vi-**

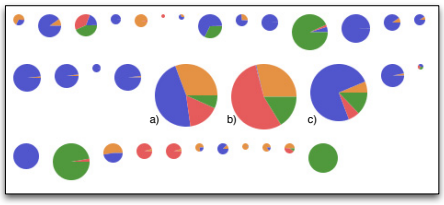
[1] http://www.itea2.org/

[2] Software Evolution, Refactoring, Improvement of Operational & Usable Systems

ESEC-FSE'09, August 24–28, 2009, Amsterdam, The Netherlands.
ACM 978-1-60558-001-2/09/08.



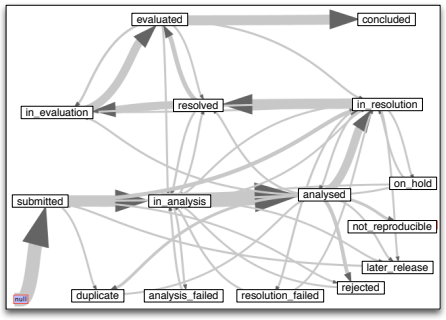
1: Polymetric view of PRs grouped by analyzer



2: Pie chart to visualize process step lengths

sualization. The size (*i.e.*, the area) of the pie is mapped to the total time from the creation of the PR until it was closed. The colored pie parts are mapped to the four process steps: *submitted* (orange), *in_analysis* (blue), *in_resolution* (red), *in_evaluation* (green).

Figure 2 shows an example of a pie chart view where each pie represents a PR. In this view, besides the obvious differences in the size of the circles, one can also see, that there are big differences in the relative durations of the process steps. There are also PRs that are missing certain phases, or the phases are to short, in comparison to the others, so that they are not visible. Focusing on the three biggest charts in the middle (labels: a, b, c), we can see, that a) was almost half of the time *in_analysis*, b) on the other hand has a very small *in_analysis* phase and c) has a relatively short delay before work started after the submission of the report (only a small orange wedge).



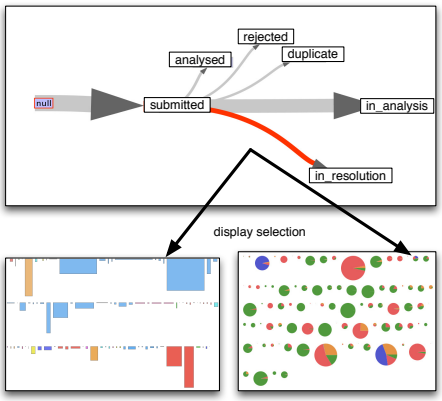
3: PR life-cycle view

With the addition of a **PR state transition view** (Figure 3) we provide a new angle and starting point for an analysis. In this view configuration, all occurring state transitions are displayed in an aggregated form. The width of the arrows between the states is mapped to the number of problem reports that exhibit the corresponding transition.

Mainly for outlier detection but also to incorporate more information, such as severity and priority, we also provide a **scatter plot view**, which can be configured to display arbitrary attributes of the problem reports (not shown here).

3. VIEW COMBINATIONS

Although we can use the presented visualization building blocks to gain some insights into problem report data, the real power of our approach lies in the combination of multiple easy to understand views. For example, we can select all the problem reports that went directly after submission into the resolution phase and therefore skipped the analysis phase. Displaying these PRs in a polymetric effort view and a state duration pie view, we get a view configuration that allows us to investigate the effects on effort estimation quality and process step durations. Depending on what we find, we can further drill down and examine the detailed problem report information of exceptional entities or use other views with different selections, color mappings, layouts, etc., to gain further insights.



4: Combination of 3 views

4. TOOL INFORMATION

Our tool is implemented in Groovy[3] and Java. It uses Prefuse[4] for visualization and Hibernate[5] for database connectivity. The features described in this paper are fully implemented and a future version will be made available to the community.

5. REFERENCES

[1] M. D'Ambros, M. Lanza, and M. Pinzger. "a bug's life" - visualizing a bug database. In *Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis)*, pages 113–120. IEEE CS Press, June 2007.

[2] C. A. Halverson, J. B. Ellis, C. Danis, and W. A. Kellogg. Designing task visualizations to support the coordination of work in software development. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 39–48, New York, NY, USA, 2006. ACM.

[3] P. Knab, M. Pinzger, B. Fluri, and H. Gall. Interactvie views for analyzing problem reports. To appear in Proceedings of the 2009 ICSM.

[4] M. Lanza and S. Ducasse. Polymetric views — a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, September 2003.

[3] http://groovy.codehaus.org/
[4] http://prefuse.org
[5] https://www.hibernate.org/

TUD-SERG-2009-030
ISSN 1872-5392