UEDashboard: Awareness of Unusual Events in Commit Histories

Larissa Leite, Christoph Treude, Fernando Figueira Filho
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, RN, Brazil
larissa.leite@gmail.com, {ctreude,fernando}@dimap.ufrn.br

ABSTRACT

To be able to respond to source code modifications with large impact or commits that necessitate further examination, developers and managers in a software development team need to be aware of anything unusual happening in their software projects. To address this need, we introduce UEDashboard, a tool which automatically detects unusual events in a commit history based on metrics and smells, and surfaces them in an event feed. Our preliminary evaluation with a team of professional software developers showed that our conceptualization of unusual correlates with developers' perceptions of task difficulty, and that UEDashboard could be useful in supporting development meetings and for pre-commit warnings.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics

Keywords

Awareness, unusual events, commit history

1. INTRODUCTION AND MOTIVATION

Since the success of software projects largely depends on the effectiveness of communication and coordination, software development teams need to maintain awareness of different aspects ranging from overall project status and process bottlenecks to current tasks and incoming artifacts [17]. Many approaches for maintaining awareness have been proposed, including tools to provide developers with insights into the workspaces of their co-workers such as Palantír [15] and dashboards for high-level status information [17].

When we asked GitHub users in a recent study about the information they would expect in a summary of development activity, *unusual events* stood out as something developers want to be kept aware of [16]. Participants mentioned several examples of such events: *"If a developer hasn't committed anything in a while, his first commit after a long silence could be particularly interesting, for example, because it took him a long time to fix a bug. Also, important commits might have unusual commit messages, for example including smileys, lots of exclamation marks or something like that. Basically something indicating that the developer was emotional about that particular commit."* Another developer added: *"Changes to files that haven't been changed in a long time or changes to a large number of files, a large number of deletions."* While there is previous research on automatically detecting buggy commits (e.g., [5, 10]), little work has been conducted on detecting unusual events in a commit history. Visualizations of the software process [8], the commit history [18], or specific commits [3] can help developers spot unusual events, but they have not been designed specifically for this purpose and their usefulness in unusual event detection has not been evaluated. Awareness of unusual events can be useful in preventing errors, and in alerting developers and managers of events that may require justification or that can affect the work of other developers, especially when the events relate to significant changes to the project.

In this work, we are introducing the Unusual Events Dashboard (UEDashboard), a tool which automatically detects unusual events in a commit history and surfaces them in an event feed. To do so, we first need to establish what is "usual" in a given context, and then identify derivations. Context is important since what is unusual depends on factors such as team size, work dynamics, software process, development cycle, domain, product size, criticality, and development model. UEDashboard takes context into account by collecting commit-related metrics, such as size, complexity, and time since last commit, as well as a number of commit smells [9], and by considering the mean value for these metrics per developer as the "usual" value. "Unusual" values are detected if they differ from the mean value by more than two standard deviations.

We conducted a preliminary evaluation of UEDashboard by showing usual and unusual commits detected by the tool in the commit history of a Brazilian software company to six of the company's developers. We found our conceptualization of unusual to correlate with developers' perception of difficulty, and we received positive feedback regarding the usefulness of UEDashboard in supporting development meetings and for pre-commit warnings.

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08...
http://dx.doi.org/10.1145/2786805.2803184

2. UEDASHBOARD

UEDashboard collects data from the commit history of a project, focusing on different unusual events detailed below. Unless otherwise noted, commits are flagged as unusual if their value for at least one metric is different from the mean for the developer doing the commit by more than two standard deviations. While the number of standard deviations is easily configurable in UEDashboard, we chose a value of two as the default configuration since in a normal distribution, less than 5% of all data points are more than two standard deviations away from the mean. Although we did not verify the normal distribution of all variables, a ratio of roughly 1 in 20 events being classified as unusual conformed with our intention of these events being unusual yet not completely uncommon. A more detailed description of the events is available in a workshop paper [13] and a thesis [12].

Time between commits is considered an indicator of project activity [11]. UEDashboard analyzes the time between commits from the perspective of a single developer and from the perspective of an entire team. While a long time between commits from an entire team might indicate infrastructure issues, for a single developer, it can indicate task difficulty and can potentially lead to merge conflicts.

Time between commits to a file can indicate problems with particular files. Files that have not been modified in a long time can indicate that code is stable or has been "forgotten" and is not up to date with the current status of requirements, architecture, and design. UEDashboard detects this event separately for each file in a project.

Number of files added, deleted, and modified are different indicators of a commit's impact. Adding a considerable amount of files might be a sign of developing a new requirement while a large number of deletions might signify changes to the software architecture or to the project requirements. A high value for the number of files modified could be caused by a refactoring or the development of a disruptive task, i.e., a task that changes a lot and "disrupts" the current code. UEDashboard treats these three metrics separately and can detect unusual events for each one.

Number of lines of code added, deleted, and modified are also indicators of a commit's impact and measures of its size. We expect the deletion of lines to be less common than their addition or modification. UEDashboard detects unusual events separately for these three metrics.

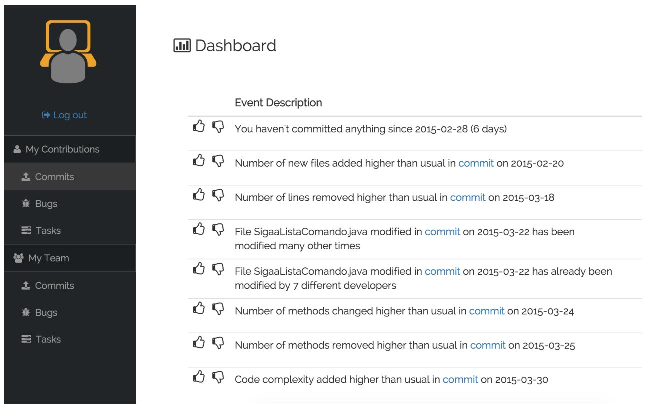
Aggregate change in complexity is a metric to characterize a commit. UEDashboard computes the McCabe complexity of all methods touched in a commit before and after the commit, and calculates the sum of the complexity changes for each commit.

Number of methods added and modified are metrics where a high value might indicate the development of new functionality. Similar to a large number of files modified, a large value for number of methods modified might signify a change that "disrupts" the current code.

In addition to these metrics, UEDashboard detects a number of "commit smells" [9] which do not follow our conceptualization of unusual events:

Files changed by many different developers are more likely to contain post-release vulnerabilities [14]. Using the mean number of developers that have changed a file as the usual value in a project and two standard deviations as the threshold for an unusual number, UEDashboard notifies developers of files that have been changed by an unusually large number of developers.

Many modifications in a single file during the lifetime of a project can indicate the presence of faults [6]. Again, UEDashboard uses the mean number of modifications made to a file as the usual value, and treats all those modifications as unusual that exceed a modification count of the mean plus two standard deviations per file.



	Event Description
👍 👎	You haven't committed anything since 2015-02-28 (6 days)
👍 👎	Number of new files added higher than usual in commit on 2015-02-20
👍 👎	Number of lines removed higher than usual in commit on 2015-03-18
👍 👎	File SigaaListaComando.java modified in commit on 2015-03-22 has been modified many other times
👍 👎	File SigaaListaComando.java modified in commit on 2015-03-22 has already been modified by 7 different developers
👍 👎	Number of methods changed higher than usual in commit on 2015-03-24
👍 👎	Number of methods removed higher than usual in commit on 2015-03-25
👍 👎	Code complexity added higher than usual in commit on 2015-03-30

Figure 1: Developer event feed

Figure 1 shows the user interface of UEDashboard, a web application written in Python using the Flask web framework. The design was inspired by our previous work on dashboards and feeds [17]. Although the main content provided by the dashboard is the feed of unusual events shown in Figure 1, UEDashboard also includes graphs, for example illustrating the number of commits made by each developer in the last few months. In addition to the feed of personal events shown in Figure 1, UEDashboard provides a feed of all unusual events in a team. Events can be filtered by date, commit, developer, and type. For each commit, detailed information can be seen in a modal window.

For each event, the user can provide feedback through up-votes and down-votes to indicate whether a given event is relevant to them. In addition, each event has a feedback button (not shown in the figure for space reasons) through which users can give written feedback for an event, such as a justification as to why a certain commit was unusual. In future iterations, we plan for UEDashboard to learn from this feedback to improve the detection of unusual events. The current implementation of UEDashboard supports SVN repositories, and we are working on releasing a version for Git in the near future.

3. PRELIMINARY EVALUATION

We conducted a preliminary evaluation of the unusual events detected by UEDashboard at Superintendência de Informática (SINFO), a company responsible for the development and maintenance of all information systems used by employees, students, and faculty at a university. More than 30 institutions in Brazil rely on SINFO's technology. Our evaluation was conducted with the team responsible for the academic procedures at a university, consisting of approximately 15 developers.

We applied UEDashboard to the 215 commits made by the development team under study in March 2015, and we used additional data from the commits between August 2014 and February 2015 for the calculation of historic means and standard deviations. Table 1 shows the results. Except for one type of unusual event (files not modified in a long time), all events were detected at least 3 times, with the two commit smells (files modified by many different developers, files modified many times) being the most frequent. Out of 215

Table 1: Occurrence count of different events

event	count
long time between commits	13
files not modified in a long time	0
large number of added files	10
large number of deleted files	3
large number of modified files	17
large number of lines of code added	8
large number of lines of code deleted	21
large number of lines of code modified	14
high code complexity	10
low code complexity	3
large number of added methods	19
large number of modified methods	10
files modified by many different developers	43
files modified many times	65

Table 2: Unusual vs. usual events

criterion	mean unusual	mean usual	p-value
difficulty	3.3	2.1	**0.0139**
criticality	3.3	3.6	0.4841
typicality	3.1	4.1	0.0606

commits, 98 (46%) were detected as unusual by at least one of the methods for detecting unusual events (59 or 27% if we disregard the two methods for detecting commit smells). While these numbers seem relatively high, no type of event occurred in more than 10% of the commits, except for the commit smells. In future work, we will prioritize these events based on user feedback.

To evaluate the idea of detecting unusual events in commit histories, we conducted semi-structured interviews with five developers and one manager at SINFO. We presented each participant with four of their own commits and the corresponding issues from the company's issue tracking system: two commits UEDashboard had classified as unusual and two commits UEDashboard had not classified as unusual. For each of these commits, we asked participants to rate the corresponding task in terms of how (i) difficult, (ii) critical, and (iii) typical it was on a five-point Likert scale. Participants were not told which commits UEDashboard had detected as being unusual. In addition, we asked all participants open-ended questions regarding the usefulness of the approach.

Table 2 shows the results of participants rating commits and the corresponding tasks regarding difficulty, criticality, and typicality. The results show that developers found unusual commits to be significantly more difficult than usual commits (Mann-Whitney test, $p < 0.05$), whereas there was no statistically significant difference for the other criteria. While developers might be aware of their own difficult tasks, UEDashboard can automatically alert them of difficult tasks their colleagues are working on.

The general feedback from the participants was positive, as this example quote shows: *"It makes sense, it would be useful for the team to understand how the code is evolving, who is changing what, what classes are more modified than others. It could also help enhance planning and could potentially increase code quality."* An example of a commit detected as unusual by UEDashboard was described by the manager: *"This commit is not very common. I was looking closely to this task when the developer was working on it and I remember it was more difficult than usual."* Mirroring the findings from our previous work [16], developers explained that unusual events require justification: *"Perhaps if a developer takes very long in one task the events can be used as justification as to why it took a long time and how complex or big the change was."*

Participants also commented on the different methods for detecting unusual events: *"Number of files modified, added or deleted. If it is far away from the average it might indicate that the solution is not good enough, there might be some unnecessary changes or something may be missing."* Another developer added: *"Number of methods added, but also looking at the task. Sometimes a task is simple enough and it needs only one method to be added, and the developer could be complicating things by adding more."*

4. EXAMPLE SCENARIOS

The major use case for UEDashboard that emerged from our interviews is that of a discussion starter or a meeting agenda. As one developer explained: *"It would be useful to be aware of unusual events from other developers. Since my tasks are related to bug fixing, the more information I have about past commits, the better. If I notice a strange modification or many modifications I can promptly talk to the developer about it."* Unusual events of new members on the team might be particularly interesting: *"The information about cyclomatic complexity is very interesting. I also find it very useful to know how long it has been since the last commit from a developer. As a manager, it is a way to look closer to what newcomers are doing, and if it takes a long time for a newcomer to make a commit, it can indicate that they are accumulating a lot of changes before a commit or that the task is too complex for them. The information would be useful in the meetings, since I could question and talk to them about their tasks without being too passive and waiting for them to tell me something."* Another developer added: *"It is good for everyone to know what the others are doing. We have meetings twice a week, but we don't go into details and we don't have the information the tool provides."*

Another scenario is using the unusual event detection to notify developers when they are about to make a commit UEDashboard would flag as unusual. We asked about the usefulness of such a feature, and one developer responded: *"Yes. It could be an indicator that something might not be right. It can also be used as justification of something that the manager could eventually question."*

5. RELATED WORK

Awareness is defined as "an understanding of the activities of others, which provide context for your own activity" [4]. Many tools have been developed to support awareness in software development, including Palantír [15] which provides insight into workspaces of other developers, and FastDASH [1] which uses a representation of a shared code base to highlight current activities aggregated at the level of files and methods.

To the best of our knowledge, no awareness tool exists that focuses specifically on unusual events in a commit history. However, there is a body of work on the detection of buggy commits. Kim et al. [10] used a machine learning classifier to determine whether a new software change is more similar to prior buggy changes or clean changes. Eyolfson et al. [5] found commits submitted between midnight and 4am to be significantly more bug-prone than those submitted at other times, and daily-committing developers to produce less buggy commits. Our goal with UEDashboard is not the detection of buggy commits, but the detection of commits that differ from what a normal commit in a given context looks like. Visualizations, such as Recovered Unified Process Views [8], visualizations of the change history [18], or Commit 2.0 [3], can support the identification of unusual events in a commit history, but they consider fewer dimensions compared to UEDashboard and do not provide an explicit feed of unusual events.

Conflict awareness tools such as Crystal [2] or WeCode [7] detect and display a particular kind of unusual event in a commit history. Crystal, for example, detects if a developer has not committed for a long time and if a developer has made changes that conflict with other developers' changes, break the build, or lead to test failures. WeCode identifies the outcomes of merging all developers' code at once. UEDashboard's definition of "unusual" is wider, and its general approach can be applied to artifacts other than source code or commits.

6. CONCLUSION AND FUTURE WORK

We introduced UEDashboard, an awareness tool for software development teams featuring an event feed of unusual events in a project's commit history. In addition to a number of commit smells, unusual events are detected when at least one metric of a developer's commit differs from the historic mean for the same developer by more than two standard deviations. Our preliminary evaluation shows that our conceptualization of unusual correlates with developers' perception of task difficulty, and that the output of UEDashboard could be used for monitoring new developers, for pre-commit warnings, and as an agenda for discussions and meetings. The ultimate goal of UEDashboard is to improve development processes through the avoidance of collaboration conflicts by making developers aware of unusual events in their software repositories.

The first step of our future work lies in expanding the detection of unusual events to other development artifacts beyond commits, in particular issues and pull requests. This will also enable us to draw on connections between these artifacts, e.g., a high number of added files in a commit related to a new use case might be expected, but the same event would be unusual if it occurred in a commit related to a bug fix. Similarly, taking into consideration the development cycle of a project, work on new features might be common at the beginning of an iteration but unusual towards the end of it. We are currently applying UEDashboard to other projects to conduct a more thorough evaluation of the idea that development teams benefit from awareness of anything unusual happening in their repositories.

7. ACKNOWLEDGEMENTS

This work is partially supported by CNPq Jovens Talentos grant 407455/2013-2, CNPq Universal grant 460904/2014-0, and CAPES/PROAP.

8. REFERENCES

[1] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson. Fastdash: A visual dashboard for fostering awareness in software teams. In *Proc. of the Conf. on Human Factors in Computing Systems*, pages 1313–1322, 2007.

[2] Y. Brun, R. Holmes, M. Ernst, and D. Notkin. Early detection of collaboration conflicts and risks. *IEEE Trans. on Softw. Eng.*, 39(10):1358–1375, 2013.

[3] M. D'Ambros, M. Lanza, and R. Robbes. Commit 2.0. In *Proc. of the Intl. Workshop on Web 2.0 for Softw. Eng.*, pages 14–19, 2010.

[4] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proc. of the Conf. on Computer-supported Cooperative Work*, pages 107–114, 1992.

[5] J. Eyolfson, L. Tan, and P. Lam. Do time of day and developer experience affect commit bugginess? In *Proc. of the Working Conf. on Mining Softw. Repositories*, pages 153–162, 2011.

[6] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy. Predicting fault incidence using software change history. *IEEE Trans. on Softw. Eng.*, 26(7):653–661, 2000.

[7] M. Guimarães and A. Silva. Improving early detection of software merge conflicts. In *Proc. of the Intl. Conf. on Softw. Eng.*, pages 342–352, 2012.

[8] A. Hindle, M. W. Godfrey, and R. C. Holt. Software process recovery using recovered unified process views. In *Proc. of the Intl. Conf. on Softw. Maintenance*, pages 1–10, 2010.

[9] S. Johnson and Z. Welch. Bad commit smells, 2013.

[10] S. Kim, E. J. Whitehead, Jr., and Y. Zhang. Classifying software changes: Clean or buggy? *IEEE Trans. on Softw. Eng.*, 34(2):181–196, 2008.

[11] C. Kolassa, D. Riehle, and M. A. Salim. The empirical commit frequency distribution of open source projects. In *Proc. of the Intl. Symp. on Open Collaboration*, pages 18:1–18:8, 2013.

[12] L. Leite. An automatic approach to detect and notify development teams of unusual events in software repositories, 2015. B.Sc. thesis.

[13] L. Leite, C. Treude, and F. Figueira Filho. An automatic approach to detect unusual events in software repositories. In *Proc. of the Latin American School on Softw. Eng.*, 2015. To appear.

[14] A. Meneely. *Investigating the relationship between developer collaboration and software security.* PhD thesis, North Carolina State University, 2011.

[15] A. Sarma, Z. Noroozi, and A. van der Hoek. Palantír: Raising awareness among configuration management workspaces. In *Proc. of the Intl. Conf. on Softw. Eng.*, pages 444–454, 2003.

[16] C. Treude, F. Figueira Filho, and U. Kulesza. Summarizing and measuring development activity. In *Proc. of the Europ. Softw. Eng. Conf. & the Symp. on the Foundations of Softw. Eng.*, 2015. To appear.

[17] C. Treude and M.-A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proc. of the Intl. Conf. on Softw. Eng. - Vol. 1*, pages 365–374, 2010.

[18] F. Van Rysselberghe and S. Demeyer. Studying software evolution information by visualizing the change history. In *Proc. of the Intl. Conf. on Softw. Maintenance*, pages 328–337, 2004.