From: "Harald Gall and Nenad Medvidovic" <icse2011-papers-chairs@borbala.com>
Subject: ICSE 2011 Paper Notification [285]
Date: 17 November, 2010 4:03:08 PM EST
To: brun@cs.washington.edu, rtholmes@cs.uwaterloo.ca, mernst@cs.washington.edu, notkin@cs.washington.edu
Cc: icse2011-papers-chairs@borbala.com, icse2011-papers-webadmin@borbala.com
Reply-To: icse2011-papers-chairs@borbala.com

Dear Yuriy, Reid, Michael and David,

Thank you for your submission to ICSE 2011. The program committee met on November 12-13, 2010, to consider the submissions to the Research Paper track. We regret to inform you that your paper,

"Proactive Detection of Collaboration Conflicts"

has not been accepted for inclusion in the conference program. The competition was very strong: only 62 of the 441 submissions were accepted, giving an acceptance rate of 14%.

We enclose below the reviewers' comments to your submission, which we hope you will find useful for your future research work.

Nevertheless, we do hope that you will be able to attend the conference, which will be held May 21-28, 2011, in Waikiki, Honolulu, Hawaii.

Please note that submission to some of the satellite events is still open (see http://2011.icse-conferences.org/content/submissions)

Sincerely,

Nenad Medvidovic and Harald Gall
Program Committee Co-Chairs
ICSE 2011

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=

First reviewer's review:

Summary of the submission <<<

This paper proposes a an approach to permit the early detection of possible merge conflicts in version control systems. The commit histories of 10 projects from github are analyzed, focusing on inconsistencies and conflicts. A tool, called Crystal, is proposed, which builds upon Mercurial to warn developers about possible conflicts.

| | | Evaluation <<<

The introduction discusses the problems of syncrhonizing too late and too early, but the paper actually is only about avoiding too late check ins.

In Figure 1, it might be useful to show Melinda's view as well, also since that view is used in the discussion in 3.5.

Figure 2 contains useful terminology: in your paper you also talk about peers and masters, which you may also want to include. Actually, the discussion of 3.5 really is specific for distributed version control systems. That's fine, but the section could be more explicit about it.

Your use of "Threats to validity" is non standard. The normal usage of this term is to walk through specific types of validity threats that apply to the research method adopted. For example, given the questions in 4.1, you might wonder whether your cases are representative (external validity), wether the statistical tools you use are appropriate (internal validity, is the conflict relationship sufficiently close to a normal distribution making it meaningful to discuss standard deviations?), do your measurements actually help to answer your research question (construct validity), and so on.

Especially the use of "threats to validity" to discuss limitations of a tool in section 5 is not very common: the better approach would be to separate tool design and implementation (new 5.1) tool evaluation with clear aims (finding possibilities and limitations of the tool, new 5.2), and threats to validity of the evaluation (new 5.3, external, internal, etc.)

In your present approach, the Crystal tool isn't really evaluated, but illustrated through an example.

I was somewhat to surprised to see a tool for Mercurial in a paper
extensively analyzing github.

Section 6 came as a surprise to me. It looks like a future work section, but it
already contains some "evidence" that this future work is going to improve the
tool. It is not entirely clear why the authors described the high-order
conflicts idea in such detail when it is not yet implementer, nor part of the
contributions of the paper. I would leave it until a future paper myself.

The data in figure 8 is very interesting -- I was very surprised
about the halting percentage: I'd be interested to see an example
of that (here's a threat to external validity: I doubt this will be
as common in other cases).

Points in favour or against <<<

+ Topic of practical relevance
+ Interesting idea
+ Tool available
+ Interesting empirical data supporting need for approach
+ Very well written paper
- More specific to distributed version management than claimed.
- Weak evaluation of the tool itself

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*

Second reviewer's review:

Summary of the submission <<<

The authors present an approach to proactively detect SCM-style merge conflicts
before the corresponding pieces of code are committed to the SCM. They motivate
their work through a study of how often conflict situation arise in non-trivial
collaborative projects. They then present their prototype implementation named
Crystal.

Evaluation <<<

This is a well-written paper with a number of issues which prevent publication
in the current form.

One issue is that the paper is trying to do two things at the same time, on the
one hand the evaluation of Crystal's potential, which however doesn't really

have anything to do with Crystal. It's more a reflection on commit and conflict policies in non-trivial collaboratively developed software systems. There is related work on that but the authors do not really mention anything, probably also because of lack of space.

The Second issue is that the authors are oblivious to one strongly related piece of work, the work on Syde by Hattori and Lanza, presented e.g. at MSR 2009 and ICSE 2010. While Syde, as far as I know, is tailored towards Java and Eclipse, it does exactly what Crystal intends to do (although Crystal wants to be language-agnostic). I do think there's quite a strong overlap there which should be considered by the authors.

The last issue is that Crystal is a prototype at best, i.e., the authors talk about its potential and its philosophy in the first 7 pages, and about its future on page 8 and 9, but Crystal itself is not evaluated at all. As such this is a paper about intentions, a bit too premature for ICSE prime time. But, there is a lot of potential in all this. Maybe a paper for the NIER track?

||| Points in favour or against <<<

+ well written
- neat, but not novel idea
- only intentions so far, no facts

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*

Third reviewer's review:

||| Summary of the submission <<<

The paper presents Crystal, a tool that can inform developers when their changes are out of synch in a Distributed CM system (e.g., Mercurial). The paper also presents a study of 8/4 OSS projects using Git that shows the number of merge conflict and synchronization issues that these projects had as a motivation for the use of Crystal. Crystal has also been deployed to a small developer group (although this information is very sketchy in the paper).

||| Evaluation <<<

The paper makes two contributions. First, it extends current work on workspace awareness that work on centralized CM systems to work in a DSCM/ DVCS era, where git and mercurial are becoming really popular. These systems bring finer grained control and a new set of challenges. Second, the authors actually perform archival analysis to drive home the point that merge conflicts occur systematically in the selected projects and that even when syntactic merges pass, there are compilation and behavioral problems.

Good related work. Although, the authors should make sure to have read/ reference the paper (The Promises and Perils of Mining Git. 6th IEEE International Working Conference on Mining Software Repositories (MSR '09), pp 1-10.), especially given that the authors are mining Git for their study.

The paper was, however, very difficult to read and in my opinion would be greatly helped with a complete rewrite. The authors should start with describing how DVCS differ from their centralized counterparts. Even after reading the paper I was not sure what a check-point was. This seemed to be a very critical part of the paper and not understanding it completely made it difficult for me to appreciate the logic of determining the conflicting changes. The authors also mention that DVCS allows for constraints that a code cannot be uploaded/committed/pushed until it passes a test case. These are new and need explanation. The table in Figure 2 helps a little, but needs better explanation.

Section 3 was particularly hard to follow, with very short and abrupt sections.

The authors provide impressive figures about number of days that a merge conflict was existing, however, given that this is OSS do the authors have any idea about the activity levels, did the contributor find a problem and get back to it in the next 5 minutes, next day, 5 days? Even if the authors don't have the information, this should be mentioned in the Threats to Validity.

The authors mention that they tested for compilation and behavioral issues for successful merges using test suites. Where did they find these test suites? Some information about these test suites and how much effort (time & computation) was involved in running these test suites will be useful.

Github provides some interesting, but very clean visualization of changes across repositories, I would have liked to see a comparison of Crystal's view with GitHub.

I was confused with the author's claims that Crystal is able to identify changes that will have compilation or behavioral issues. As based on my understanding Crystal follows a set of changes as evidenced by checkpoints and publish actions, how these changes actually integrate without causing semantic conflicts seemed to be future work in Section 6, but was claimed as functionality in related work: "Our approach, on the other hand....- textual, syntactic, and behavioral", please explain.

Minor comments:
- What is NCSL?
- Please reference tabular data as tables. It is very confusing when tabular data is referenced as figures.
- probably makes sense to have a single Threats to Validity section at the end, instead of two times. Also, the second T to V is really tool limitations.

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*