Your SBES 2016 - Technical Research paper 155202

SBES 2016 - Technical Research <esa@dcc.ufba.br> 21 de junho de 2016 11:21
Para: cesario.cristiano@gmail.com
Cc: Leonardo Murta <leomurta@gmail.com>

Dear Mr. Cristiano Cesario:

I am pleased to inform that your paper

"Workspace Awareness for Distributed Version Control Systems"

has been accepted for publication at the 30th Brazilian Symposium on Software Engineering (SBES 2016), which will be held in Maringá, PA, Brazil, from 19 to 23 September 2016.

The reviews of your paper are attached below and are also available at https://jems.sbc.org.br/PaperShow.cgi?m=155202

When preparing the camera-ready version of your paper, please be aware of the following important actions:

1. Please send an email to the PC chair (esa@dcc.ufba.br), until July 4, confirming the list of authors (full name and affiliation), in the order that should appear in the proceedings, and indicating which author will present your work. The inclusion of your work in the proceedings is conditioned to this confirmation, registration of the indicated author in CBSoft, and the actual presentation of the paper in SBES.

2. It is mandatory that the final version of your paper reflects the reviewers comments and suggestions for quality improvements, clarifications, and corrections. We will double check that you have implemented the required changes in the camera-ready version.

3. The deadline for submitting the camera-ready version is July 11, 2016. The final version should be uploaded using the JEMS system via the "Camera Ready" file upload option. Only PDF files are accepted. The maximum length is ** 10 pages ** and the ACM conference style should be strictly followed.

4. Copyright information will be sent to you after July 11, 2016.

Once again, congratulations for the paper acceptance, and I am looking forward seeing you in Maringá.

Regards,
Eduardo Almeida
SBES 2016 TPC Chair

===== Review =====

*** 1: Summary (A brief description of the paper.): This paper presents an approach for monitoring changes across cloned repositories of distributed version control systems such as Git. Although adopting a DVCS brings benefits to distributed and collaborative software development, keeping up with the changes from different clones and branches is a challenge. Therefore, the approach aims to increase

awareness of developers when working on a project with multiple contributors, by discovering and tracking commits from other clones and branches, presenting information in different detail levels. The approach was evaluated with a retrospective analysis of an existing repository, a performance analysis, as well as by an observational study that provided initial evidence that the approach/tool can be used to help on tasks conducted by repository developers and maintainers.

*** 2: Paper strength (What are the positive aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): + Increasing awareness during development can improve quality of contributed code, and even indirectly avoid conflicts + Prototype tool implemented together with the approach + Evaluation, although initial, takes into account multiple perspectives where the approach an tool can be evaluated + Relevant topic and worthy contribution

*** 3: Paper weakness (What are the negative aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): - When presenting the problem, could further elaborate on the motivation, and how problems related to awareness could be avoided with such approach/tool - Related to the above issue, assumption that any changes might be of interest to all developers - Design decisions over the different evaluations are not discussed - Time measured during the observational study is impacted by activity on phase 1

*** 4: Comments to authors (Comments and suggestions for improving the paper.): Although the proposed approach and tool is intuitively useful to developers, the paper lacks a better motivation for the problems that it could solve and/or mitigate. For instance, to introduce the need for such a tool and solution, the text could provide practical examples of the problem, and the consequences that one would have due to the absence of this tool in terms of productivity and quality losses/reductions. Perhaps real examples could be described, if any is available, or at least simple scenarios based on such real examples, illustrating what could happen during development if we do not have DyeVC. Such motivation would help to understand the need for this tool and how developers would use it during their daily activities, and could also improve the paper.

Therefore, I would suggest using a running example for illustrating the DyeVC approach. It is a bit difficult to follow Section 2.1 without a concrete example, at first. Later it becomes clearer.

Also, an issue that I would expect to be discussed is the assumption that any changes in a peers repository are interesting to the developer. It could be the case that DyeVC reports a great deal of information, since cloned repositories might have a large number of commits to be synchronised. However, the differences may not be of particular interest to the developer in its current task, since they might touch unrelated files. Is this something that DyeVC also intends to tackle, maybe consider some form of ranking or prioritisation over the provided information to be visualized? I understand that presenting information in different levels of detail already helps, but it would be worth considering how the tool would perform in large projects, not only as done in the current analysis of time/space complexity, but also in projects with a large number of contributors, and consequently, larger number of cloned repositories.

Q1 refers to identifying which clones were created from a repository. Technically there is no way to detect which clones were created, but a possibility is to approximate this information through developer commits, as you did in the post-hoc evaluation. Another possibility is to register such clones, as discussed in Sec. 2.1. However, it is unclear how does one registers clones, who is responsible for performing such task and maintaining current information up to date?

Regarding the post-hoc evaluation: Why forks were not used to do this post-hoc? I assume they would provide a similar view on what the tool intends to do, since they are basically clones, but in many cases people create forks as development branches. The current post-hoc evaluation only focuses on what has been integrated into trunk, which is a subset of the activity that actually occurred in each developers cloned repository.

On the subject of the observational study, the time measured on phase 2 is biased given that participants had some knowledge of the scenario. (also see comment below from the additional reviewer)

About the performance evaluation, how many repository clones are taken into account in the analysis? I believe that the time and space can also be affected by the number of cloned repositories. Also, the text could briefly mention possible improvements to that, given that we should not have to consider the whole history of a repository to perform synchronisation.

Also, a further evaluation in the future could be a qualitative study focused on professionals that work as project maintainers (administrators as the text mentions) and developers.

Below I include further detailed comments from the additional reviewer (some have also been mentioned above):

On section 2.2.2 -> there are four tracked commits ready to be pulled. -> what do you mean by ready"?

On section 2.2.3 -> Furthermore, it is uncommon to have a scenario where pushes are performed from a developer to another -> is there any empirical evidence to support such claim?

On figure 7, why commit a2bd8 does not have thicker borders since it is a branchs head?

On section 4.1 -> Although one might perform a merge manually and insert a different text in the comment, this did not compromise our analysis because we had a focus on depicting some of the merge situations, and not all of them. -> Why did you had such focus, and what kinds of merge situations are not taken into account?

There is a serious limitation regarding the time metric collected during the observational study since participants used the same scenario to answer the questions using both approaches, one after the other. On phase 2 (when they used DyeVC to answer the questions), they could have answered faster because of previous knowledge on the scenario. Although the authors recognise such threat, they do not mention any strategy to mitigate it.

A suggestion to improve DyeVCs performance would be to compute changes considering a short period of the development history (considering the last 6 or 3 months changes, for example), since developers would be more interested in keeping up with more recent changes.

A suggestion to improve DyeVC's usability would be to separate different projects notifications into different tabs, instead of showing notifications for all projects in a single screen, as shown in figure 5.

Minor issues: - In the last paragraph of Sec. 1 there is the message Erro! Fonte de referência não encontrada. - Increase font and resolution of some figures

Regarding the rebuttal: - I suggest making clear the usage scenario of DyeVC in the final version, to explicitly detail how the tool would automatically register the clone, and so on. - If there is space, I would also suggest to include the comment on why forks were not considered in the evaluation. - In its current form, the text gave me

the impression that adding more clones would (negatively) affect DyeVC's performance, so the comment on the rebuttal should also be included in the text.

*** 5: Questions for rebuttal phase (Provide direct questions that you want answers during the rebuttal phase. Remember that authors will have only 4000 characters to answer all questions from all reviewers, so add here only the most relevant questions.): Is there any possibility for filtering, prioritising or ranking the visualised information?

Who is responsible for registering and managing clones?

Why forks were not used to do this post-hoc?

About the performance evaluation, how many repository clones are taken into account in the analysis?

===== Review =====

*** 1: Summary (A brief description of the paper.): The paper describes the design, implementation and preliminary evaluation of a system, called DyeVC, to support extraction and visualization of dependencies among software repository clones.

*** 2: Paper strength (What are the positive aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The paper is generally well written and structured.

The proposed system is clearly described in terms of its main design decisions and implementation choices.

The evaluation provided solid evidence that the system can effectively and efficiently achieve its design goals.

*** 3: Paper weakness (What are the negative aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The title could be improved to better reflect the work described in the paper.

The motivation behind the work is not adequately discussed.

The evaluation goals seem a bit contrived towards the kinds of topological information provided by the proposed tool.

*** 4: Comments to authors (Comments and suggestions for improving the paper.): First of all, the term "workspace awareness", which is used in the paper's title, is not explained nor even mentioned in the rest of the paper. What does "workspace" mean in this context? I believe the term "topology awareness" would be a better fit for the title considering what's described in the paper.

Secondly,the paper falls short of adequately motivating the need for the kind of topology-related information provided by the proposed system. In particular, it's not clear from the paper how simply providing a topological overview indicating which commits belong to which clones (in contrast to, for instance, indicating potential conflicts between them) would actually facilitate or improve the software development process in a DVCS setting. In this regard, the authors could give more concrete examples of possible problems that development teams could face when working with DVCS clones, and of how such problems could be better addressed or even eliminated with the help of the kinds of information provided by DyeVC.

Finally, with the lack of a better motivation, it seems that the three

evaluation questions are a bit biased towards the exact kinds of topological information that can be obtained with the help of the proposed tool. In this regard, it's no surprise that the tool was successful in supporting developers answering all of them. Where do those three questions come from? How were they defined? These issues should be better justified in the paper in order to support the authors' claim that the proposed tool can be a valid contribution to assist developers in managing the evolution of DVCS clones.

Minor comments and corrections:

- The acronym SCCS is used by not defined anywhere in the paper (sec. 1)

- Cross-reference errors (sec. 1)

- "Section ... shows the evaluation" - > "Section ... *describes* the evaluation" (sec. 1)

- "there is no sense in giving him" -> "there is no sense in giving *it*" (sec. 2.2.3)

- "the commit N will be located in the right hand side of commit N-1" -> "the commit N will be located *to the right* of commit N-1"

- "This way" -> "In this way" (several places)

- Labels are too small and barely legible in some figures (e.g., Figs. 4, 7, 12)

*** 5: Questions for rebuttal phase (Provide direct questions that you want answers during the rebuttal phase. Remember that authors will have only 4000 characters to answer all questions from all reviewers, so add here only the most relevant questions.): See comments to authors.

===== Review =====

*** 1: Summary (A brief description of the paper.): The paper presents DyeVC, a tool that raises developers' and repository administrators' awareness about activity on clones of their repositories by means of notification, topology visualization, detailed information on tracked branches, and a combined visualization of commits in all clones. The authors evaluate the tool with actual open source projects by means of a post-hoc study, an observational study, and a performance study.

*** 2: Paper strength (What are the positive aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The paper is very well written, detailed, and technically sound.

*** 3: Paper weakness (What are the negative aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The visualization aspect could be explored further (see "Comments to authors").

*** 4: Comments to authors (Comments and suggestions for improving the paper.): Some aspects of DyeVC as an interactive visualization tool could be explored further:

- How is the navigation between the different levels of detail? Is it possible to select a node in Level 2 and then switch to Level 3 focusing on that node's branches? (When thinking of DyeVC as a visualization tool, it may be helpful to remember Shneiderman's "visual information seeking mantra": Overview first, zoom and filter, then details-on-demand.) - In Level 2 (Topology), the number of not-synced commits is shown as text; it could also be mapped to a visual attribute,

such as line thickness. This way, users could get a glimpse of repository activity.

Regarding the performance evaluation, the authors focus on correlations and barely discuss the actual values of the time metric. In particular, the time needed to display the commit history seems too high. Was this a problem for users in the observational study? What can be made to reduce these times?

Although Table 5 helps understand the relationship between execution time and input (# commits, size, and # files), a multiple regression analysis would be more informative.

Minor:

- Page 2, "including a more *thoroughly* discussion" => *thorough* - Page 2, "Erro! Fonte de referência não encontrada" - Page 6, "but it is *public* available" => *publicly* - Page 9, "if conflicts are *willing* to occur" => *likely*

*** 5: Questions for rebuttal phase (Provide direct questions that you want answers during the rebuttal phase. Remember that authors will have only 4000 characters to answer all questions from all reviewers, so add here only the most relevant questions.): - How is the navigation between the different levels of detail? - Did the participants perceive the waiting time for the commit history as long? What can be made to reduce this time?

===== Review =====

*** 1: Summary (A brief description of the paper.): This papers proposes and evaluates a visual approach, namely DyeVC, to assist developers in identifying and analyzing clones among distributed source code repositories.

*** 2: Paper strength (What are the positive aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The paper deals with an emergent problem of Distributed Version Control Systems (DVCSs), that any software development project with a distributed team faces. In fact, repository clones may increase very fast and many of the them become obsolete over time. These obsoletes clones harms the repository management seriously.

The problem and motivation are well presented in the Introduction section. The approach implementation is also properly presented. The authors provided a fair evaluation of their approach using open source projects for post-hoc analysis and an observational study with four subjects. There is also a performance evaluation. The related literature is sufficient.

The proposed approach is perfectly applicable and indeed helps developers in managing the repository clones.

I would like to highlight that the approach implementation was carefully presented, with enough details for researchers that may want to further investigate this topic. Furthermore, and more important, it presents an appropriate evaluation of the proposed approach.

*** 3: Paper weakness (What are the negative aspects of this paper, specially with respect to Originality and novelty; Technical soundness and contribution; Readability, organization and presentation; Comparison to previous work; Evaluation.): The observational study can be improved. The number of subjects is very small. In addition, I think it is missing an analysis of the experience of the subjects. As I could observe, P3 took considerably less time to perform his tasks and DyeVC seems to be not a factor for his performance. What was the reason for that? Is P3

much more experienced than others? In order to improve your paper, you must provide this analysis. You can do this in 1 or 2 paragraphs.

*** 4: Comments to authors (Comments and suggestions for improving the paper.): This is a good work. Congratulations! Please, consider improving the observational study, mainly in regard to the subject experience.