Indirect Conflicts: A Research Proposal

Jordan Ell
University of Victoria, Victoria, British Columbia, Canada
jell@uvic.ca

I. Proposal

As Software Configuration Management (SCM) systems have grown over the years, their maturity has allowed parallel development practices to become the norm in software development. With this parallel development comes the need for larger awareness among developers to have "an understanding of the activities of other which provides a context for one's own activities" [1]. This added awareness mitigates some negative aspects of parallel development which include the cost of conflict prevention and resolution, however, in practice we see these mitigated losses continue to exist and occur frequently. Indirect conflicts are one of the parallel development downsides in that as once developer changes code inside a project, it may affect another developer's code at a different location within the same code base or to an external code base. Many researchers, myself included, have taken on the task to study and create awareness mechanisms for software developers in order to help them either prevent, catch, or resolve indirect conflicts over the lifetime of a project. However, the resulting tools of academic pursuit have attracted little interest from developers and have had many common issues such as information overload, high false positive rates [2] [3] [4] [5].

This research aims at creating an understanding as to why our mitigation attempts in indirect conflict prevention and resolution have not had as large a success as was initially thought to occur by understanding the nature of indirect conflicts, studying how real world developers currently handle the prevention or resolution of indirect conflicts, and by looking at current tools used. These new understandings will help us fill in the gaps presented by previous research in this area. In my research methodology, I will use expert software developer experience and opinion to analyze what has become the downfall of indirect conflicts mitigation tools as well as the state of the research surrounding indirect conflicts. While many researchers have conducted small sample case studies of their indirect conflict research, few have produced follow up academic papers as to why repetitive problems continue to occur in the body of work. By conducting semi structured interviews, I hope to fill this gap with a large body of developer insight and opinion. In order to understand how indirect conflict issues occur across the whole spectrum of software development, I have chosen to conduct interviews from a mix of software companies which differ from waterfall development, agile development, API creation, software as a service, open source, closed source, databases management, and more. This wide spectrum allows this research to become highly generalizable and applicable not only to academics, but to real world industry professionals.

In the early stages of my research (Summer 2013) regarding indirect conflicts, I have conducted and analyzed interviews with 19 industry professionals from a large collection of software companies which includes IBM, Microsoft, and Amazon. Findings have begun to indicate a disjoint between the academic study of indirect conflicts and the real world application of techniques and practices to prevent and resolve such conflicts. These finding have primarily indicated that developers approach indirect conflicts with a curative thought process, being that of only dealing with issues after they arise, oppose to a preventative measure which is more sought after in academia, being that of preventing indirect conflicts before they occur at the risk of preventing conflicts which may never arise. In the continuing steps in my research, I will conduct two larger studies. The first will be to continue the exploration of the ideas of prevention versus cure methodologies in regards to indirect conflicts. By conducting large controlled experiments supplemented by qualitative developer interviews, I hope to discover how and when prevention techniques or curative techniques may be better than the other. Second, I will create an awareness mechanism which takes the form of a methodology, practice, or development tool which will be best suited towards developer's needs and the results found in the previous research. I will validate this mechanism through industry officials using and examining the final product for its merits in the software development industry.

This proposed research has significant impact, not only on academia, but on industry as well. Through the exploration of indirect conflicts, we will provide researchers with a clear outline for what has been done wrong in the past, but also as to how we should, as an academic community, focus our efforts moving forward to better serve those which will benefit from indirect conflict research (industry). These efforts, as anticipated, will be the study of prevention versus cure techniques for indirect conflict management. Industry will benefit from the resulting processes or tools to come out of this research, in that they will know better how to manage their time effectively by preventing or curing indirect conflicts. Industry should see better production from software developers, as well as an overall improvement in software quality.

References

[1] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, ser. CSCW '92. New York, NY, USA: ACM, 1992, pp. 107–114.

[2] A. Sarma, G. Bortis, and A. van der Hoek, "Towards supporting awareness of indirect conflicts across software configuration management workspaces," in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ser. ASE '07. New York, NY, USA: ACM, 2007, pp. 94–103.

[3] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, "Fastdash: a visual dashboard for fostering awareness in software teams," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 1313–1322.

[4] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 23–33.

[5] F. Servant, J. A. Jones, and A. van der Hoek, "Casi: preventing indirect conflicts through a live visualization," in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '10. New York, NY, USA: ACM, 2010, pp. 39–46.