Enabling Workspace Awareness for Collaborative Software Modeling

Jae young Bang, Daniel Popescu, Nenad Medvidovic
University of Southern California
Los Angeles, California, USA
{jaeyounb, dpopescu, neno}@usc.edu

ABSTRACT
Distributed software modeling is common today, although geographically separated designers need to overcome several communication challenges. Software designers typically use version control systems (VCSs) to integrate their work. However, existing VCSs do not continuously inform all designers of new design decisions and conflicts. Designers often introduce conflicts precisely because they are unaware of such design decisions. Research on collaborative implementation has explored workspace awareness to deal with this challenge, and observed that providing workspace awareness facilitates conflict detection and resolution. However, existing workspace awareness tools typically do not work well as-is for collaborative modeling. We envision the emergence of new types of collaborative modeling tools that provide various forms of workspace awareness.

INTRODUCTION
Today, economic advantages encourage the development of software involving geographically-distributed stakeholders [15]. For example, a client residing in Europe may work with a software development company in America, which outsources the work to Asia. Further, software development teams themselves are increasingly geographically distributed. Although distributed software development has its benefits, communication challenges between geographically separated team members typically dilute these benefits [9, 10, 12].

Software modeling is an important activity for designing and documenting a software system [16]. To support coordination between distributed software designers, version control systems (VCSs) for collaborative modeling have been developed and employed [1]. However, even state-of-the-art VCSs introduce a delay between the times when local design decisions are made and when developers incorporate others' decisions. Consequently, a designer is often unaware of design decisions that made by others and introduces conflicts that could be expensive to resolve.

The negative consequences caused by the lack of such "collaborative awareness" are not unique to collaborative software modeling. In fact, they have been studied more prominently in the case of collaborative software implementation. *Workspace awareness* is "the up-to-the-minute knowledge of other participants' interactions with the shared workspace" [8]. Existing collaborative software implementation tools that enable workspace awareness are proactive; they continuously provide information about the changes made to the code base, in order to allow the distributed programmers to detect and react to potential conflicts [3, 5, 6, 14, 17]. Empirical data has shown that enabling workspace awareness helps conflict detection and resolution [13].

Unfortunately, existing workspace awareness tools typically do not work well as-is for collaborative modeling. Many of them do not understand the rich syntax and semantics of software models. Software designers will need additional support to achieve modeling-specific workspace awareness.

We posit that, since collaborative implementation benefits from workspace awareness, so will collaborative modeling. We envision two types of workspace awareness tools that can aid collaborative modeling: (1) tools that support traditional copy-edit-merge to encourage parallel design activities and also inform the designers of other designers' activities and newly occurring conflicts, and (2) tools that allow designers to work together in a single synchronized workspace. Another promising research direction is to apply speculative analysis [4] to predict and prevent conflicts in advance. Overall, such collaborative tools would allow distributed designers to coordinate more frequently (or continuously) in order to avoid conflicts.

In this paper, we discuss (1) the limitations of existing collaborative modeling tools, (2) why existing techniques that focus on collaborative implementation are insufficient, and (3) our ideas for enabling software modeling workspace awareness.

DRAWBACKS OF CURRENT SOFTWARE MODEL VCS
Today, physically separated software designers typically use VCSs to exchange their design decisions, to be aware of each others' intentions, and to avoid conflicts. However, those tools do not update the designers continuously with the changes made to the model. As a result, the designers may, and do, subsequently make conflicting design decisions.

Altmanninger et al. surveyed and divided existing VCSs into two sets based on how the VCSs share design artifacts: pessimistic and optimistic approaches [1]. A pessimistic approach uses locking to prevent multiple designers from modifying the same part of the system at once. The alternative to this is the optimistic approach, in which each designer maintains her/his own local copy of the model, makes design decisions in parallel, and merges them later. This approach is also called copy-edit-merge.

The pessimistic approach has two principal drawbacks: simple locking techniques cannot prevent indirect conflicts that involve dependencies between artifacts, and locking slows down collaborative modeling activities. Simple locking techniques that do not consider dependencies are unable to completely prevent conflicts. For example, one designer could lock and modify a module that depends on a library. If another designer locks and modifies that library at the same time, these modifications may conflict. Moreover, even advanced locking techniques that do consider dependencies and automatically preempt actions on all system parts that depend on the part being edited are too restrictive to the designers because they prevent parallel work and slow down the design process, which tends to be free-flowing.

The major drawback of the optimistic collaborative modeling approach is that the designers are not informed regularly of the design decisions that other designers are making. The existing tools that implement the optimistic approach merge new design decisions either only when requested by the designers or periodically. That results in longer time for a design decision to travel from the moment of its creation to the perception at the other end. We call this time *design decision latency*. High design decision latency often increases the frequency of conflicting design decisions.

Furthermore, the designers are notified of newly occurring conflicts only when a merge and retrieval of design decisions or a specific query of new conflicts are requested. If the designers do not know of existing conflicts, it is likely that they will consequently make critical design decisions causing new conflicts that are even more difficult to resolve. The cost to detect and resolve such conflicts can be prohibitive, especially when they are left undetected for long periods. The cost becomes even worse when the wrong design decisions have been propagated into the implemented system.

APPLICATION OF WORKSPACE AWARENESS

The drawbacks of existing VCSs for modeling are, of course, not unique to collaborative software modeling; collaborative software implementation also shares similar challenges. To cope with these challenges, previous research explored providing workspace awareness for collaborative software implementation. This research has resulted in tools that provide information about changes being made and potential conflicts to all affected distributed developers [3, 5, 6, 14, 17]. These tools have shown that achieving workspace awareness helps developers to observe and resolve conflicts earlier.

However, existing collaborative implementation tools are not intended and typically do not work well as-is for collaborative modeling. Many collaborative implementation tools use line-based merging to coordinate new changes [11]. Those tools may allow merging that invalidates the model because they neglect the syntax and semantics of the software model. The tools that utilize more sophisticated merging may work for collaborative modeling, yet it is still an open question how the design-level models can be mapped into appropriate internal representations that can be captured by these tools.

A modeling-specific workspace awareness tool will have to resolve two types of conflicts: synchronization and higher-order conflicts. *Synchronization conflicts* are conflicts that prevent the local instances of a software model to be aggregated into a single consistent model. They occur when multiple designers modify the same software modeling artifact or closely related artifacts. *Higher-order conflicts* are conflicts that violate the syntax of the modeling notation or the intended semantics of the model. Higher-order conflicts are harder to detect than synchronization conflicts as they often require computationally expensive model checking algorithms.

The existing collaborative implementation tools may not support certain aspects of the software design process. In collaborative software modeling, software designers can make modeling changes in parallel from many different views of the model, resulting in higher-order conflicts that occur across views. Also, software models that utilize customized meta-models (e.g., domain-specific models) require tailored tools for conflict detection and resolution. As a consequence, reusing existing workspace awareness tools would require higher-order conflict resolution support built on top of these existing tools.

A LOOK TO THE FUTURE

Software modeling is typically a highly-collaborative activity in which possible design solutions are explored in a team of designers. The way distributed designers collaborate determines the type of necessary tool support. We envision two types of collaborative modeling tools that enable workspace awareness: (1) the traditional copy-edit-merge design process, which facilitates exploring individually design solutions and enables workspace awareness for distributed software modeling teams, and (2) realtime synchronization, which enables workspace awareness for teams that work simultaneously on the same model artifacts.

Workspace awareness can be provided to software designers who do not have overlapping work hours. It is common that a software development team works round-the-clock (or follow-the-sun) [15]. A software designer in such a team is unaware of the design decisions that have been made overnight while s/he is away. By providing an aggregated rather than complete, raw list of design activities that have been made by other designers while the designer is away, the returning designer would be made aware of the current status of the model. This should help to prevent making conflicting design decisions when s/he resumes working.

Another way to provide workspace awareness is to implement a single synchronized workspace through realtime synchronization. The goal of realtime synchronization is enabling a distributed collaborative modeling environment that has nearzero decision latency as it would be the case for collocated software designers. To realize such an environment, coordination techniques such as operational transformation [7] can be used. Synchronization conflicts can be automatically resolved based on the operation types (creation, removal, or modification of an artifact) or policies such as designers' rankings. For example, a design decision made by a designer could be automatically rejected if it conflicts with a decision made by a higher-ranked designer. Such automatic synchronization helps each designer to maintain an up-to-date copy of the model.

An interesting opportunity is presented by speculative analysis. Recent work [4] has demonstrated how speculative analysis can be used to predict proactively conflicts during collaborative implementation. Proactive collaborative modeling tools could provide even a higher degree of workspace awareness by predicting potential conflicts before they are made. For example, a proactive collaborative modeling tool could monitor on which parts of the model each participating designer is working and warn designers who are modifying closely related artifacts about potential overlapping modifications before the decisions are made (e.g., via speculative analysis).

We ultimately envision hybrid collaborative modeling environments that provide configurable degrees of workspace awareness. In such hybrid collaborative modeling environments, software designers are able to freely choose the amount and type of information that they want to receive.

Overall, enabling workspace awareness will help geographically distributed software designers to coordinate with less burden when dealing with conflicts.

ACKNOWLEDGEMENTS

This work has been supported by Infosys Technologies Ltd.

BIOGRAPHY

Jae young Bang

Jae young Bang is a second year PhD student at the University of Southern California. His thesis topic is in collaborative software modeling.

Daniel Popescu

Daniel Popescu received a PhD in computer science from the University of Southern California. His research interests include software architecture, program comprehension and distributed event-based systems.

Nenad Medvidovic

Nenad Medvidovic is a professor in the Department of Computer Science at the University of Southern California and director of the USC Center for Systems and Software Engineering. Medvidovic received a PhD in information and computer science from the University of California, Irvine.

REFERENCES

1. Altmanninger, K., Seidl, M., and Wimmer, M. A Survey on Model Versioning Approaches. *IJWIS* (2009).
2. Bang, J., Popescu, D., Edwards, G., Medvidovic, N., Kulkarni, N., Rama, G., and Padmanabhuni, S. CoDesign – A Highly Extensible Collaborative Software Modeling Framework. *Proc. ICSE Tool Demo* (2010).
3. Biehl, J. T., Czerwinski, M., Smith, G., and Robertson, G. G. FASTDash: a visual dashboard for fostering awareness in software teams. *Proc. CHI* (2007).
4. Brun, Y., Holmes, R., Ernst, M., and Notkin, D. Speculative Analysis: Exploring Future Development States of Software. *FoSER* (2010).
5. Brun, Y., Holmes, R., Ernst, M., and Notkin, D. Proactive detection of collaboration conflicts. *Proc. ESEC/FSE* (2011).
6. Dewan, P., and Hegde, R. Semi-Synchronous Conflict Detection and Resolution in Asynchronous Software Development. *Proc. ECSCW* (2007).
7. Ellis, C., and Gibbs, S. Concurrency control in groupware systems. *Proc. SIGMOD* (1989).
8. Gutwin, C., and Greenberg, S. Workspace Awareness for Groupware. *Proc. CHI* (1996).
9. Herbsleb, J. Global software engineering: The future of socio-technical coordination. *Proc. FoSE* (2007).
10. Hinds, P., and Bailey, D. Out of sight, out of sync: Understanding conflict in distributed teams. *Organization Science 14* (2003).
11. Mens, T. A State-of-the-art Survey on Software Merging. *TSE* (2002).
12. Olson, G., and Olson, J. Distance matters. *Human-Computer Interaction 15* (2000).
13. Sarma, A., Redmiles, D., and van der Hoek, A. Empirical Evidence of the Benefits of Workspace Awareness in Software Configuration Management. *Proc. FSE* (2008), 113–123.
14. Sarma, A., Redmiles, D., and van der Hoek, A. Palantír: Early Detection of Development Conflicts Arising from Parallel Code Changes. *TSE* (2011).
15. Sengupta, B., Chandra, S., and Sinha, V. A research agenda for distributed software development. *Proc. ICSE* (2006).
16. Taylor, R., Medvidovic, N., and Dashofy, E. *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, 2008.
17. Wloka, J., Ryder, B., Tip, F., and Ren, X. Safe-commit Analysis to Facilitate Team Software Development. *Proc. ICSE* (2009).