the replacement of an annotation is analog to stepping into the definition of a function call. The only difference is that the method body entered during debugging is associated to an annotation and not to a function call.

Notification Bypassing: The Knockout-specific notification mechanism is established through *observable* functions that allow inspecting model elements and thus, this mechanism is exploited by the KCA to capture model manipulations. Individual model properties (e.g. task name or task due date) can only be monitored if they are declared as Knockout observables. The Knockout framework offers three methods to declare observables: observable(), computed() and observableArray(). While the observable method allows declaring simple model properties (e.g. name), the computed method can declare aggregated properties (e.g. first and last name) and observableArray is used to declare arrays. However, if developers circumvent this Knockout-specific notification mechanism, the KCA has no means to record model manipulations and the sync mechanism breaks.

Runtime Model Enhancements: Once the kca.js script is loaded by the browser, the included annotation processor locates annotations and replaces them with JavaScript code that accommodates the listener and the replay logic. If the model definition changes at runtime (e.g. a task object is enhanced with a new *priority* property), the KCA will not take notice and will fail syncing this novel model property. Constantly examining all runtime objects for property enhancements could eliminate this limitation but at the cost of performance degradation. Since changing the model definition at runtime is rather exceptional, we did not adapt the current KCA implementation.

7. CONCLUSION

Incorporating shared editing capabilities in web applications using traditional concurrency control libraries is a time-consuming and tedious task. Therefore, we evaluated the ColADA solution which promised to increase development productivity since lightweight source code annotations are leveraged instead of using conventional collaboration libraries that induce the need to rigorously change the application's source code.

Through the transformation of the widely-adopted Knockout framework into a collaboration-enabled web application framework, we showed that source code annotations are in the first place a viable option to introduce collaboration features. Moreover, a developer study employing the adapted Knockout framework could affirm our hypothesis that the annotation-based ColADA approach can outperform a traditional concurrency control library. In the particular developer study, we compared the enriched Knockout framework with the SAP Gravity library and the results showed that the development time as well as the required source code changes can be substantially reduced when adopting an annotation-based solution. Additionally, the developer study exhibits that programmers are generally more satisfied with an annotation-based approach when comparing software quality characteristics like functional suitability, compatibility, usability, maintainability and satisfaction. Besides being beneficial for development efficiency, annotations are also a capable means to define multiple product variants (e.g. single-user and multi-user version) in one single code branch.

8. ACKNOWLEDGMENTS

This work was partially supported by funds from the European Commission (project OMELETTE, contract number 257635).

9. REFERENCES

[1] *ISO/IEC FDIS 25010 : 2010 (E) - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models*. 2012.

[2] The Apache Software Foundation. Apache Wave. `http://incubator.apache.org/wave/`, 2012.

[3] Agustina, F. Liu, S. Xia, H. Shen, and C. Sun. CoMaya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools. In *Proc. CSCW*, pages 5–8, 2008.

[4] F. Buschmann, K. Henney, and D. C. Schmidt. *Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. John Wiley & Sons, 2007.

[5] A. H. Davis, C. Sun, and J. Lu. Generalizing Operational Transformation to the Standard General Markup Language. In *Proc. CSCW*, pages 58–67, 2002.

[6] C. A. Ellis and S. J. Gibbs. Concurrency Control in Groupware Systems. In *Proc. SIGMOD*, pages 399–407. ACM, 1989.

[7] J. Gentle. ShareJS - Live Concurrent Editing in your App. `http://sharejs.org/`, 2012.

[8] M. Heinrich, F. Lehmann, T. Springer, and M. Gaedke. Exploiting Single-User Web Applications for Shared Editing - A Generic Transformation Approach. In *WWW*, pages 1057–1066, 2012.

[9] A. Prakash and M. J. Knister. A Framework for Undoing Actions in Collaborative Systems. *ACM Trans. Comput.-Hum. Interact.*, 1:295–330, 1994.

[10] A. Rickayzen. Collaborative Process Modeling. `http://scn.sap.com/community/bpm/business-process-modeling/blog/2012/03/20/`, 2012.

[11] S. Sanderson. Knockout : Home. `http://knockoutjs.com/`, 2012.

[12] UI Development Toolkit for HTML5 Developer Center. `http://scn.sap.com/community/developer-center/front-end`, 2012.

[13] H. Shen and C. Sun. Flexible Notification for Collaborative Systems. In *Proc. CSCW*, pages 77–86, 2002.

[14] G. Simpson, A. Roe, and R. Lewontin. *Quantitative Zoology: Revised Edition*. Dover Books on Biology, Psychology and Medicine. Dover Publications, 2003.

[15] C. Sun, S. Xia, D. Sun, D. Chen, H. Shen, and W. Cai. Transparent Adaptation of Single-User Applications for Multi-User Real-Time Collaboration. *ACM Trans. Comput.-Hum. Interact.*, 13:531–582, 2006.

[16] S. Xia, D. Sun, C. Sun, D. Chen, and H. Shen. Leveraging Single-user Applications for Multi-user Collaboration: the CoWord Approach. In *CSCW*, pages 162–171, 2004.

[17] N. C. Zakas. *Professional JavaScript for Web Developers*. Wrox, 2012.