Syncro - Concurrent Editing Library for Google Wave

Michael Goderbauer, Markus Goetz, Alexander Grosskopf,
Andreas Meyer, and Mathias Weske

Hasso-Plattner-Institute, Potsdam 14482, Germany

Abstract. The web accelerated the way people collaborate globally distributed. With Google Wave, a rich and extensible real-time collaboration platform is becoming available to a large audience. Google implements an operational transformation (OT) approach to resolve conflicting concurrent edits. However, the OT interface is not available for developers of Wave feature extensions, such as collaborative model editors. Therefore, programmers have to implement their own conflict management solution.

This paper presents our lightweight library called syncro. Syncro addresses the problem in a general fashion and can be used for Wave gadget programming as well as for other collaboration platforms that need to maintain a common distributed state.

1 Introduction

Collaboration is the basis for joint value creation. With increasing complexity of tasks to conduct in business, science, and administration, efficient team collaboration has become a crucial success factor.

Software tools for collaboration typically support the exchange of data and centralized storage of information, for instance in shared workspaces. Only in recent years, technology became available allowing real-time collaboration via the web-browser. This enables team partners to edit a document concurrently and let others see changes instantly.

Google Wave is a new technology that promises to bring real-time collaboration to a large audience. A wave is a hosted conversation that allows multiple users to work on a text artifact at the same time. Editing conflicts are resolved using operational transformation (OT) [1]. Furthermore, a wave's functionality can be extended by gadgets. Even though gadgets live inside of waves, they are not per se collaborative because the OT interface implemented by Google is not available to third-party developers. For that reason, gadget programmers have to find their own answer to handle conflicting edits.

In Section 2, we discuss OT approaches as underlying concept for conflict resolution. We explain the syncro library, which allows conflict resolution in Google Wave gadgets in Section 3. Implementation details, demo-gadgets and further work are discussed in Section 4 before we conclude the paper in Section 5.

B. Benatallah et al. (Eds.): ICWE 2010, LNCS 6189, pp. 510–513, 2010.

2 Operational Transformation for Conflict Resolution

When multiple users edit a single artifact at the same time, conflicts are inevitable and therefore, sophisticated algorithms are needed to handle those conflicting edits. A very popular text-based approach to do this is operational transformation, which is implemented in several collaborative text editing tools, such as MoonEdit[1] or Etherpad[2]. OT relies on Lamport's notion of total ordering [2] to define a unique sequence of events in a distributed system (more details in Section 3). Moreover, OT performs transformations to apply remote modifications, such as insert or delete, on the local client [1]. In addition to the mentioned online text editing tools, this approach was also implemented in a generic OT library for Windows [3]. Linked with the code, the library turns most single-user desktop applications, such as Word, PowerPoint, or Maya, into a collaborative version.

Following the trend towards online collaboration, Google has also implemented OT for collaborative text editing inside Google Wave. However, as their OT interface is not available inside of gadgets, programmers have to implement their own conflict management. One possible approach would be the reimplementation of OT for gadgets. This would require serializing the artifact edited in the gadget to text before OT transformations are applied. From an engineering point of view the indirection of converting artifacts to text is far too complicated. A more convenient way would be to apply the modifications directly to the artifact.

3 Syncro - A Distributed Command Stack for Google Wave

Syncro is a library that provides a generally applicable solution for concurrent editing conflicts of complex artifacts in platforms like Google Wave. It's based on the command pattern [4], a software design pattern in which each modification to an artifact is encapsulated in a command object. Using this concept, syncro shares the commands among all participants of a wave.

Since Google Wave gadgets have no interface to send or receive commands, syncro stores this information in the gadget state, a key value store synchronized among all participants of a wave. To make sure that no command is overwritten, each command has to be stored at a unique key.

Furthermore, to assure that each participant has the same view on the edited artifact, the commands need to be applied in the same consistent order on all clients. The challenge of ordering events in a distributed system has been addressed by Leslie Lamport in his paper from 1978 [2]. He introduces an algorithm for synchronizing a distributed system of logical clocks, which can be used to order events consistently among all participants. These logical clocks are called Lamport clocks and their value is increased whenever an event occurs. In Google Wave the events are the commands created independently by each participant. Lamport's algorithm attaches a logical clock value and a unique sender ID to each command to establish a strict total order among all commands. Therefore,

[1] `http://moonedit.com/`
[2] `http://etherpad.com/`

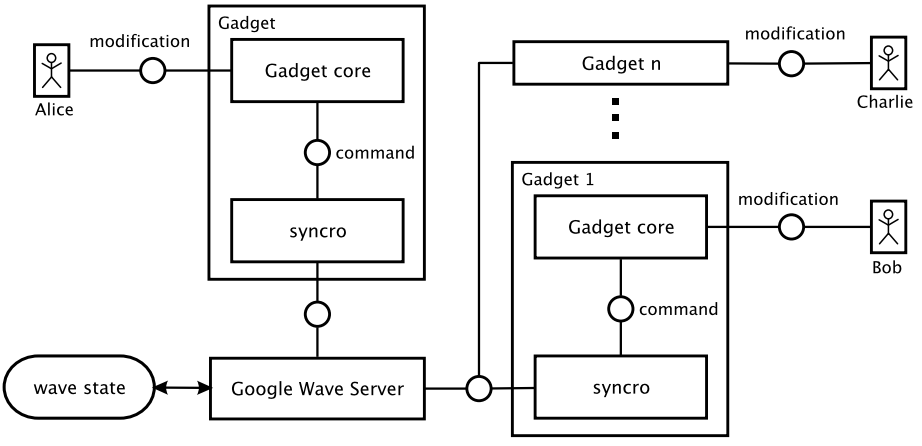

Fig. 1. Architecture of syncro

each combination of logical clock value and sender ID is unique. In our context, we can use the globally unique Google Wave user ID (e. g. alice@googlewave.com) as sender ID.

Based on this, we implemented a JavaScript library named syncro. It can be used by gadget programmers to solve the problems related to concurrent editing. The system architecture of syncro is shown in Figure 1. When Alice interacts with the gadget, each interaction creates a command. The gadget then pushes the command to its local syncro instance. Syncro attaches the current Lamport timestamp consisting of the clock value and Alice's Google Wave user ID to the command. Afterwards, the extended command is stored in the gadget state using its Lamport timestamp as unique key. This state modification invokes a callback in the syncro instances of the other Wave participants, here Bob and Charlie. The callback function retrieves the new command from the wave state and inserts it chronologically correct into the local command stack using the attached Lamport timestamp. Syncro then informs the gadget about the new command.

Obviously, Bob can also interact with his gadget while Alice's command is still in transmission. When Alice's command arrives at Bob's client and syncro decides that her command happened before Bob's, the syncro library calls the gadget with the commands to be reverted and the commands to be applied. In our case, the gadget reverts Bob's command before applying Alice's. Afterwards, Bob's command is reapplied. This guarantees that each gadget applies the commands in exactly the same order ensuring that all wave participants have the same view of the edited artifact in the gadget.

4 Library Implementation and Applications

The library presented in Section 3, the source code and a demo application is publicly available[3]. The demo shows the functionality of a simple collaborative

[3] `http://bitbucket.org/processwave/syncro/`

graphic editing tool and can be tried out in a public wave[4]. More technical details and further links can be found on our blog[5]. For programmers, syncro solves the problem of managing conflicting edits in Google Wave gadgets and its underlying concepts can be used in a wide variety of collaboration platforms. We implemented syncro as a framework component for a collaborative business process modeling tool we are integrating into Google Wave. Our modeling gadget will build on the open source project Oryx[6], a model editor running inside the browser. Oryx supports various process modeling languages (e.g. BPMN, EPCs, PetriNets), UML Class diagrams, xForms editing and much more. For example, Figure 1 was modeled with Oryx. A Wave-aware version of Oryx using syncro is planned by June 2010 and we will demonstrate it at the conference.

5 Conclusion

This paper addresses the challenge of editing artifacts collaboratively in Google Wave gadgets. To solve the conflicts related to those concurrent edits, we have developed a lightweight JavaScript library named syncro. Syncro is based on Lamport clocks [2] and programmers implementing the command pattern can use the library to enable collaborative editing. For us, syncro is a base technology to integrate the Oryx model editor into Google Wave.

Acknowledgements

We thank the processWave.org team for their development and design activities. Namely, Michael Goderbauer, Markus Goetz, Marvin Killing, Martin Kreichgauer, Martin Krueger, Christian Ress and Thomas Zimmermann.

References

1. Sun, C., Ellis, C.: Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, pp. 59–68. ACM, New York (1998)
2. Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM 21(7) (1978)
3. Sun, C., Xia, S., Sun, D., Chen, D., Shen, H., Cai, W.: Transparent Adaptation of Single-User Applications for Multi-User Real-Time Collaboration. ACM Trans. Comput.-Hum. Interact 13(4), 531–582 (2006)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Abstraction and Reuse of Object-Oriented Design. In: Nierstrasz, O. (ed.) ECOOP 1993. LNCS, vol. 707, pp. 406–431. Springer, Heidelberg (1993)

[4] http://tinyurl.com/demowave
[5] http://www.processwave.org/
[6] http://www.oryx-project.org/