How does Google Wave fit in the Context of a Web-Based Enterprise Collaboration Platform?

Philipp André Heinemann

Technische Universität München
Chair of Software Engineering for Business Information Systems
Boltzmannstr. 3
85748 Garching bei München

Abstract: Web-based communication and collaboration is a significant trend in modern enterprises. However, communication as well as collaboration still makes use of various platforms like email, instant-messaging, or wikis. Google introduced *Google Wave* to solve the disadvantages in usage caused by the combination of multiple technologies. The contribution of this paper is an analysis comparing Wave to other kinds of communication and collaboration with regard to its application as an enterprise 2.0 tool. Based on this, an integration prototype is implemented, which reveals the essential effort of interconnecting Google Wave to an Enterprise Content Management System.

1 Introduction

Email was invented more than 40 years ago, but is still the most frequently used technology for communication using the internet [BZL04] . In the meanwhile, the way people communicate has evolved and team collaboration became of central importance. As a result, around 26% of knowledge workers feel that email is overused in their organization and 15% actually feel that it diminishes their productivity [Dav05].

At the Google I/O[1] in May 2009, Google first announced its new communication and collaboration tool called *Wave*. It aims to provide features of established communication technologies like email and instant messaging in combination with features of Web 2.0 technologies like wikis, blogs and fotosharing in a single tool. It was opened as a developer preview to around 6,000 people in July 2009 and released on September 30th to around 100,000 users via invitations. At the time of writing this paper, Google Wave is still a non-public beta version.

The goal of this paper is to provide a detailed comprehension of the concepts and possibilities based on a very early version of Google Wave regarding the application in an enterprise context.

It is structured as follows: Section 2 provides an overview of features as well as the architecture of Google Wave and compares it to established communication technologies. In

[1] Yearly developer conference hosted by Google

Section 3, Google Wave is then examined regarding common enterprise 2.0 functionalities and, in Section 4, regarding the feasibility of integration in a web-based collaboration platform. The paper concludes with a summary and future research proposals.

2 Google Wave

In this Chapter, Google Waves architecture and features are presented to give a basic understanding of the technology. Many features make Wave differ from common communication technologies. Google itself describes Wave by the following three statements. Thus, a Wave is . . .

. . . equal parts conversation and document. "People can communicate and work together with richly formatted text, photos, videos, maps, and more." [Goo09a] Therefore, Waves are not just messages like emails are, but objects embedding documents, extensions as well as communication. Consequently, Waves can be messages or documents or any kind of combinations.

. . . shared. "Any participant can reply anywhere in the message, edit the content and add participants at any point in the process. Then playback lets anyone rewind the Wave to see who said what and when." [Goo09a]

Replies to a Wave can be placed below a recent message, like in emails, but in case of responding to a specific part of the message, also inside the recent message without citing this particular part. Therefore, replies stay in context. Furthermore, people can be added to an existing Wave. These people need to be informed about what has happened so far in this particular Wave. Therefore, the *playback* feature provides functionality participants can use to watch recent changes being made step by step. As more people participate in a Wave, it might become necessary to have a private discussion with a specific participant. This is possible using private messages, which are only shown to the sender and selected recipients. The advantage is that this discussion remains in the flow of the Wave.

These scenarios demonstrate features used inside the *Wave Frontend*, but there are opportunities given by the *Wave Embed API* to share a Wave in any HTML*Hypertext Markup Language*-environment. This and other APIs[2] will be discussed later in this paper.

. . . live. "With live transmission as you type, participants on a Wave can have faster conversations, see edits and interact with extensions in real-time." [Goo09a]

That means, if two people participating in the same Wave are online at the same time, one can see the other ones typing transmitted character by character. Equally when someone is editing the content, his name appears where he is typing, to point out where changes are being made. In future releases, the user will be able to switch this feature off, for example when writing sensitive content.

[2]Application program interface

Additionally, Google Wave provides features that do not fit into this characterization. These are listed in some blogposts, like for example [Sie09] by Siegler.

In contrast to Gmail or other Google applications, Wave is going to be open sourced. In combination with its APIs, this should lead to an active developer community enhancing Google Wave. For development, a good understanding of the architecture of Google Waves ecosystem, as well as its components' internal structure is essential. Both are presented in the following sections.

2.1 The Architecture of Google Wave Providers

In this section a coarse insight into the architecture of Wave will be given to convey the essential appreciation for subsequent content.

Comparable to email-providers, which are hosting the email-service, a *Wave Provider* is someone hosting a Wave Server. By using the Wave Federation Protocol, this *Wave Server* can interact with multiple Wave Servers, even hosted by other Wave Providers.

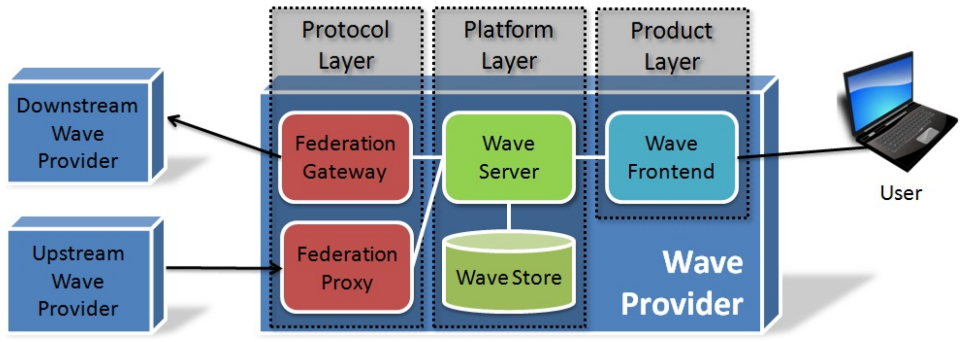

Figure 1: Wave Federation Architecture [Goo09d]

Therefore, a Wave Provider is organized in three layers as shown in Figure 1. The communication with other Wave Servers takes place in the **Protocol Layer**. This layer contains the Federation Gateway as well as the Federation Proxy, which are components of the underlying network protocol, called the Wave Federation Protocol. It realizes the sharing of Waves between multiple Wave Providers. The Federation Gateway pushes changes to associated Wave Providers whereas the Federation Proxy receives this information [Goo09d].

The Waves themselves are stored in the **Platform Layer** which is realized by the Wave Server as well as the Wave Store. Those form the basis for the Product Layer and offer a set of APIs for developers [Fer09]. These three APIs address different concerns:

1. The "Wave Embed API" enables placing Waves in HTML pages outside the Wave-Client.

2. The “Robot API” enables the implementation of extensions as robots.
3. The “Gadget API” enables the implementation of extensions as gadgets.

The **Product Layer**, which is represented by the Wave Frontend, is the client application layer providing interfaces for end users to access the Wave. The currently offered interface is a fully web-based HTML5 solution written in GWT[3] [Fer09, Goo09b].

In this paper, the platform layer is focused.

2.2 Data Model of a Wave

The platform layer especially stores Waves and makes them accessible to client products or API methods. A Wave “object” therefore consists of several entities, which are described in the following.

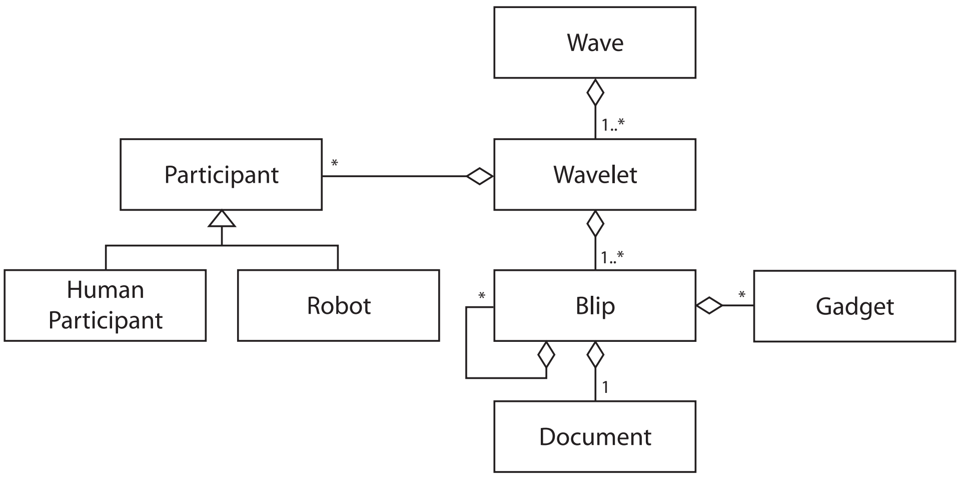

Figure 2: Entities of a Wave

Figure 2 shows the essential entities of a Wave. This structure is derived from the API released on 09/16/2009, since Google has not published a comparable Figure yet and the textual description on [Goo09b] is not exactly mapping its API.

The data model is starting with the entity Wave. A **Wave** serves as a container for one or more Wavelets, as well as the state of the Wave and historical information [Goo09b]. **Wavelets** are threaded conversations that manage participants and serve as containers for the content. The **participants** can be humans or robots, which are automated participants. The content of a Wavelet consists of one or more Blips, which also can form a hierarchical structure. This content-structure is initiated by a so called Root-Blip, which at least has a

[3]Google Web Toolkit

document, but can have following blips. A **Blip** therefore, is the basic unit in a conversation, while the contained **Document** is the message itself – for example a written text.

Google Wave offers two kinds of extensions. On the one hand **Robots**, which are automated participants of a Wave, that are able to read, edit, add and delete Blips, as well as manage participants and access third-party web-based services like humans are able to. And on the other hand **Gadgets**, which are small applications that can be added to a blip. The particular application instance of a gadget, for example a map or a poll, "lives" in the wave, whereby all participants use the very same instance.

Extensions play an important role in integrating Waves into third-party applications or vice versa. The use of robots in automated interactions with the content of a Wave is addressed in Section 4 of this paper.

2.3 Google Wave in Comparison to Common Technologies

In the following, common communication technologies and Google Wave are classified by synchronicity as well as directness in communication. Both are typical criteria used to consider usage scenarios for collaborative work using the internet [Dix97, Gj03].

Media synchronicity is named by Dennis et al. as "a communication [. . .] that encourages individuals to work together on the same activity, with the same information, at the same time" [DVS+98]. A face-to-face meeting or technologies like telephone and videoconferences have a high degree of synchronicity. Those are categorized as synchronous, while emails, faxes and voice-/video-mails, which have a low degree of synchronicity, are classified as asynchronous [CG04]. Equally, blogs, wikis and microblogs are typical asynchronous media.

Further, communication technologies can be classified with repect to their directness. A direct communication implies that messages are addressing a named set of receivers, while an indirect communication releases information to the public. Emails, faxes, instant messaging, telephone and video/voice-mail are typical direct communication technologies, whereas wikis, blogs, radio or television are typically indirect.

Common communication technologies can easily be classified regarding both criteria. Google Wave however cannot be classified clearly. This is because Wave acts in many different ways depending on the situation of use. In the situation two participants are online at the same time, Wave transmits written text character by character in nearly real-time, making Wave synchronous in this moment. On the other hand, when a receiver is offline, the sent message will be received asynchronously and as a whole when the participant comes online. Similarly, Wave acts directly in case participants are invited to the particular Wave. By making a Wave public, an indirect communication takes place.

While most communication technologies match one single classification in synchronicity and directness, Google Wave can match each of them and even switch during the lifetime of a particular Wave. Therefore it is a very flexible medium.

3 Wave in Enterprise 2.0

In the last years so-called Web 2.0 technologies emerged, which fosters team collaboration and knowledge exchange. The term Web 2.0 is still not clearly defined. In general, it relates to changes in technology as well as user behavior [CH07], that leads to interactive service-mashing applications focussing on user generated content, to encourage users' participation. The utilization of such tools within enterprises is called enterprise 2.0 [BMN08].

The goal of this Section is to assess Google Wave in comparison to common enterprise 2.0 tools, regarding usability in the context of enterprise 2.0. Therefore, common features of enterprise 2.0 technologies are identified and Wave as well as email, instant messaging, wikis and blogs are examined regarding these features.

3.1 Common features of Enterprise 2.0 Tools

McAfee has defined six common features of enterprise 2.0 technologies, which form the technology paradigm, comparable to what windows, icons, menus and pointers mean for WIMP user Interfaces [McA06]. These features called "SLATES" are presented in the following.

Search "For any information platform to be valuable, its users must be able to find what they are looking for." This can be enabled by page layouts and navigation aids, but fulltext searches are in favor [McA06].

In addition, McAfee names studies revealing that less than half of the professional staff reports that it is easy to find what they are looking for on their intranet, while internet searchers report successful search experiences in 87%.

Links In the internet the content is developed by many people, who use links to refer to appropriate content. This means that Enterprise 2.0 technologies have to offer their content in a linkable manner.

Authoring In the past years, blogs and wikis have shown that many people desire to author. Offering tools that affect people to do so, shifts a static intranet to a dynamic and interlinked work of many, which is constantly updated and leads to high-quality content [McA06].

Tags Tags are a way to categorize content by the users of the intranet. These simple, one-word descriptions emerge over time and lead to a categorization that does not necessarily correspond to an up-front categorization scheme created by the author, but a categorization that reflects the information structure and relationships the users actually use.

Extensions mean the ability of a system to recommend other appropriate informations, which are convenient or extending the currently visible content [BM07].

Signals In collaborative work new shared information can be created high frequently. An important notion is to support the users' awareness of activities by broadcasting notifications [TK94].

These common features form a good frame to compare various technologies. Most former researches, like [BMN08], focus the examination of whole enterprise 2.0 platforms, which are a combination of various tools. It seems that there is no recent work examining communication and collaboration instruments, like wikis or blogs, solely. Because of that some web-based communication and collaboration tools as well as Google Wave are considered and compared in the next section.

3.2 Fulfillment of the Requirements by the Media

We considered web-based communication and collaboration tools regarding SLATES. A summary of the results is shown in Table 1 and discussed in the following.

	E-Mail	*IM*	*Blog*	*Wiki*	*Wave*
Search for own obj.	Yes	Yes	Yes	Yes	Yes
Search for others' obj.	No	No	Yes	Yes	(Yes)
Links to http	Yes	Yes	Yes	Yes	Yes
Links from http	No	No	Yes	Yes	Yes
Authoring	doi	doi	doi	doi	doi
Tags	No	No	Yes	Yes	Yes
Extensions	No	No	doi	doi	doi
Signals	Yes	Yes	doi	doi	Yes

Caption: doi = depends on implementation

Table 1: Results of consideration regarding SLATES

To achieve a finer consideration, we divide **search** into two different functionalities: At first the ability to *search for a users' own objects* that at least all current implementations provide – including Google Wave. And secondly the ability to *search for other users' objects*, that means the ability to search for emails, instant messages or Waves of colleagues, which is apparently not possible in direct media without being participant of the particular conversation. In contrast, however, such a search is possible in indirect media like wikis or blogs, where the content is usually public. This is how Wave acts in the case a Wave is made public.

Likewise we divide the feature **links** into the following types: At first the ability to *link third party media in a Wave* using an http-hyperlink, which is possible in every current technology. And secondly the ability to *link a Wave in third party media*, like linking a Wave in a blog or website using a clickable hyperlink. This is typically possible in indirect media like blogs or wikis, but not in direct communications where the content is typically not public. Equally to blogs or wikis, Waves can be linked using http-hyperlinks which lead to the Wave-Client. A user can access this linked Wave in the case the user has a valid Wave-Account. If the Wave is not made public, the user has to be participant of the particular Wave to view the content.

Like McAfee [McA06] mentioned by naming **authoring** in the SLATES, enterprise 2.0 media have to enable and aid employees to author. But this is mainly dependent on effective easy to use frontends of the tools and less on the underlying technology.

Furthermore, **tags** are a popular way to organize content in modern indirect collaboration technologies. Most implementations of wikis or blogs offer this functionality to the public and so does Wave. Since some client-implementations of direct media allow users to tag conversations, these tags live only in the client application and are not shared with other participants.

McAfee also mentions **extensions**, which are a feature only found in very few blog or wiki implementations. There is no example in email or instant messaging and likewise it is not in Wave. But even if Wave does not have this feature yet, it is not excluded that an extension suggesting public Waves will be implemented in the near future.

At last, **signals** are well known in direct media like email and instant messaging, where new messages show up in the users inbox. Indirect media, however, do usually not have comparable inboxes, but this depends on the particular implementation. In Google Wave public instances are placed in the inbox of the Wave-Client equally to private Waves and can therefore signalize activities to the user.

The summarization of these facts in Table 1 shows that legacy technologies provide less of these features than novel ones do. This goes along with [Eva93], who mentiones in the example of email, that despite its flexibility it is not appropriate for all necessary types of communication.

For collaborative work, employees can choose from numerous information and knowledge management as well as communication tools [SLP09]. Some of these might fit better for a specific task than others, which in turn may be the choice for other tasks. Therefore, users or companies should develop convenient strategies for context-aware media usage [RF09].

Nevertheless, email is the major communication media in enterprises [BM07]. But from the point of view of a modern web-based enterprise 2.0 it lacks many features that appeared a long time after email was invented, while novel media seem to fulfill those common features.

In addition, users will substantially benefit from integrated collaboration platforms, which combine needed tools in an intuitive environment [Fox05]. Google Wave is a technology which could replace existing tools or add additional functionalities to an enterprise collaboration platform, which is the reason for a deeper look at its integration feasibility in the following Section.

4 Integration of Wave into a Web-Based Collaboration Platform

The preceding section shows that Wave has strength that address enterprise 2.0 concerns. In this section basic integration requirements are investigated and an integration prototype of Google Wave into a web-based collaboration platform is developed to get a realistic insight into the integration prospects of Google Wave.

4.1 Requirements for Integration

Due to the flexibility of Google Wave (shown in Section 2) it can be used in various ways and therefore, a large multiplicity of integration variants is conceivable. Some use cases of Wave, which can be reasonable causes for an integration, are pointed out in [Goo09a].

Furthermore, Adrian Mihai demonstrates that there is a connection between collaboration and communication [Mih09]. Equally, Lars Rasmussen mentioned at the Google I/O in 2009 that information is as well in documents as in communication [Goo09a]. In traditional communication tools, we often use this communication-part separated from documents, which are attached or linked to the communication. In Google Wave these borders are blurred and allow collaborative commenting and editing directly in the document – means that the communication takes place directly inside the document, and therefore keeps in context. Consequently, it can be interesting to share recorded communications as knowledge with the goal of avoiding situations in which employees struggle with problems other employees know the solution for [SLP09].

Because of numerous variants of integration szenarios, Wave has to provide API functionalities that are flexible enough to fulfill szenario-specific requirements. To give a general insight, no specific scenario will be discussed in the following, but the basis of embedding Wave in a web-based collaboration platform.

In this case a user should get the complete control over Waves right in the integrating platform, which therefore can be used as the single point of access for users in the enterprise. In this circumstance, Wave works like a background service which is incorporated into the collaboration platform that provides the frontend.

Users should be able to view a list of Waves, edit and show them, as well as create new Waves from scratch without the need of using the Wave-Client. Therefore, these functions should be accessible right in the integrating platform, without the need to login to Wave concurrently. In addition, it is typically intended to link other content in a Wave or vice versa link a Wave in other content, to mesh information and search it using the built-in search provided by the integrating platform. For a seamless integration this search must be extended so that Waves will be searched and retrieved like any other content.

To realize such an integration, both the integrating platform as well as the integrated technology, should provide appropriate functionalities.

4.2 Wave and its API evolve

At the time of this paper, Google Wave is still an early non-public preview with an incomplete API. Because of that, some of the abovemetioned functions are missing. For instance, there is no kind of "single-sign-on", which results that a user has to login concurrently into the integrating platform as well as in the Wave-Client. Equally, there is no possibility to constrain a user and his rights on a Wave, whereby all participant have the same authority.

Moreover, functionalities like creating, searching and accessing a Wave are incomplete in the current version of its API. This API consists of three parts: The *Embed API*, which makes it possible to embed Waves in HTML and therefore to display them in any web-based environment outside of the Wave-Client. The *Robot API* which makes it possible to implement extensions which can be added to a wave. And finally, the *Gadget API*, that enables the implementation of mini application, which can be embedded in Waves.

The *Embed API* is currently a simple JavaScript embedding the Wave in HTML and the Robot API is fully reactive, because robots are event listening extensions [Goo09c]. Therefore, the abovementioned integration functionalities can currently not be implemented by simply calling corresponding API functions actively out of the integrating platform. In the following, a found solution is presented.

4.3 The implemented Prototype

In this prototype we developed a robot to propagate necessary information to a web-based collaboration system called Tricia. Tricia is an open source web collaboration and knowledge management software developed by the InfoAsset AG[4]. This powerful platform provides integrated services [AG09] like Wiki collaboration, personal & team blogging, file & directory sharing as well as content publishing & site navigation.

Tricia has an exposed architecture which consists of the core, called "toro" and several plugins like "wiki" or "blog". In addition, this open architecture makes it easy to develop own plugins to extend the functionality of the platform. Especially the latter makes Tricia a good choice for this research.

For this prototype, we extended Tricia with a plugin we simply called "Wave". This plugin consists of several http-handlers which realize specific functionalities, like

- list all registered Waves (listHandler)
- display a Wave embedded in the Tricia user interface (defaultHandler)
- delete a Wave (deleteHandler)

These handlers can be called by users via buttons in the Tricia user interface.

[4]http://www.infoasset.de/

Because Waves' Embed API is a simple JavaScript, which does not offer callable methods to interact with a Wave, implementing a robot is currently the only way to access and manipulate Waves from outside. Until now, robots are exclusively reactive, what implies that manual interaction is indispensable to initiate a robot.

We implemented the following handlers, which can be called via http-requests and therefore used by Wave-Extensions to communicate with Tricia.

- register a Wave in Tricia (createHandler)
- synchronize Wave and Tricia (updateHandler)

Figure 3 shows the robot and http-handler we implemented to realize the integration.

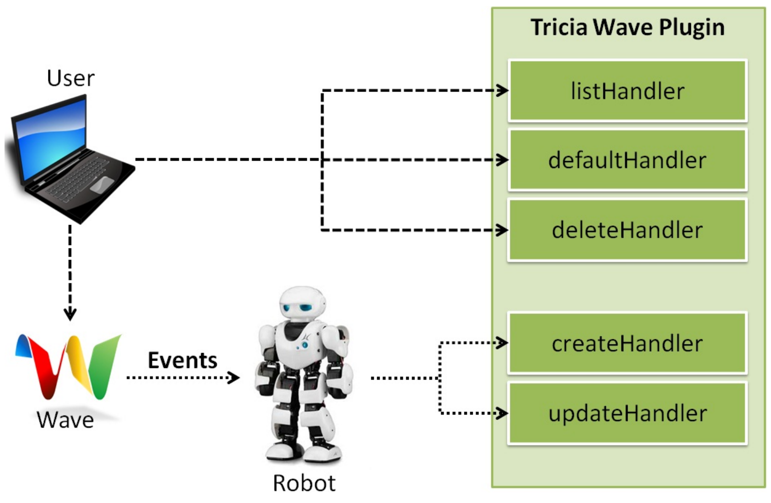

Figure 3: Schematic Realization of the Integration

Our first activity is the creation of a Wave from scratch and therefore the first connection between Tricia and Wave. Because of the passive character of robots, the following steps must be performed manually by the user.

1. Open the Wave-Client
2. Create a new Wave (and insert initial content)
3. Add the robot which connects this Wave with the integrating platform

After the completion of these steps, which inevitably have to be done manually, the robot can perform necessary communication with the integrating platform to connect to the Wave. This is possible now, because adding the robot to a Wave is the first event the robot can catch and subsequently perform further actions.

In this prototype the robot calls the Tricia *createHandler* to register a Wave in Tricia. This handler at first checks the privileges of the user who added the robot and then submits information like the waveID, time and date of creation, creator and the content. This is necessary because these information cannot be queried later. After this step the Wave is registered in Tricia and can be listed and displayed embedded in the Tricia user interface, assuming the user is logged into his Google Wave account.

Beside this initial connection, updating the metadata on changes in the Wave is an essential activity. To maintain information like the date of the last change and the content, which are stored in Tricia, the robot has to transmit the data when a change happens in the Wave. Therefore, an *updateHandler* has been implemented, which gets called by the robot to update the associated metadata in Tricia. Like mentioned before, even the content has to be transmitted and stored as metadata, because this is currently the only way to enable the Tricia-Search to find Waves matching a specific searchterm.

Next to registering and updating Waves in Tricia, the unregistration of a Wave is a necessary activity and realized by a *deleteHandler* in Tricia. After an authorized user clicks the delete-button, displayed in the Tricia user interface, the deleteHandler gets called. The challenge in this case is, that the robot is still a participant of the Wave after the deletion in Tricia and still sends updates when the content of the Wave changes. Since there is currently no way to actively remove the robots' participation, we decided to wait for the next occurring event which triggers Tricias' updateHandler and let the updateHandler return a "DENIED" message. The robot then will stop to send updates. After this, the Wave is unregistered in Tricia and the robot will no longer send updates.

But the robot is still a participant of the Wave. A future solution will be to enable the robot to remove himself on a received "DENIED" message, but currently the API method removeParticipant() has no effect. Therefore, the robot politely asks other participants of the Wave: "I can't leave self-contained. Can someone please remove me?"

5 Discussion and Conclusion

Common technologies can be clearly classified regarding synchronicity as well as directness. In contrast, Google Wave provides direct as well as indirect communication which also can be synchronous as well as asynchronous. Therefore, Wave is highly flexible, what is the basis to conceal multiple communication and collaboration needs with a single medium.

Furthermore, current direct communication technologies have weaknesses regarding common features in the enterprise 2.0 context, whereas current indirect tools, like wikis or blogs, better provide these features. This indicates that Google Wave has the potential to replace well established media, like email or instant messaging, which have been invented a long time before Web 2.0 arise.

McAfee mentioned in [McA06] that merging multiple systems in one platform, like for example the use of one huge wiki instead of many specialized and unconnected ones, has beneficial impacts on collaboration. Nevertheless, people choose specific ways of communication with respect to the relative purpose [SLP09]. Google Wave will blend these purposes of communication, if it is going to be used as a single channel for web-based communication and collaboration. Equally, merging communication channels leads to a large amount of data in a single tool and Wave has to provide features to access and organize this informaton in an extremely effective fashion. Therefore, usability might be the key factor of Waves success or failure. Future research should examine on the one hand, if users are going to adopt Google Wave this way after its public release and on the other hand, how it can fulfill related requirements.

We examined the feasibility of integrating Google Wave in web-based collaboration platforms based on the current version of Google Wave, which is the developer preview. We noticed that Wave still has lots of weaknesses in its API functionalities. But due to its possibility to extend Waves functionalities via robots, we were able to realize a showcase, as presented in Section 4. This is an indication of Waves flexibility in development.

Currently, development for Google Wave is inconvenient since there is no possibility to debug applications or even test-run them on a local machine. But the development and the APIs are getting more and more improved. For example, the Robot API got extended with the *Active Robot API*, which allows robots to act not only passively, but also actively without the need for triggering events [Cod10].

In the current version, a satisfying integration is not possible, but if Google as well as the open source community will address functions regarding users' permissions and access to Wave and its search from outside Wave Providers and robots, Google Wave could become a serious alternative to email and instant messaging.

References

[AG09] InfoAsset AG. Tricia, 2009. `http://www.infoasset.de/wikis/infoasset/home` last checked 02/15/2010.

[BM07] Jacques Bughin and James Manyika. How businesses are using Web 2.0. *McKinsey Global Survey*, 2007.

[BMN08] Thomas B, Florian Matthes, and Christian Neubert. A Concept and Service Based Analysis of Commercial and Open Source Enterprise 2.0 Tools. *International Conference on Knowledge Management and Information Sharing*, 2008.

[BZL04] Nancy K. Baym, Yan B. Zhang, and Mei-Chen Lin. Social Interactions Across Media: Interpersonal Communication on the Internet, Telephone and Face-to-Face. *New Media & Society*, 6(3):299–318, 2004.

[CG04] John R. Carlson and Joey F. George. Media Appropriateness in the Conduct and Discovery of Deceptive Communication: The Relative Influence of Richness and Synchronicity. *Group Decision and Negotiation*, 13(2):191–210, März 2004.

[CH07] Petra Cyganski and Berthold H Hass. Potenziale sozialer Netzwerke für Unternehmen. *Web 2.0. Neue Perspektiven für Marketing und Medien*, pages 101–120, 2007.

[Cod10] Google Code. Introducing Robots API v2: The Rise of Active Robots, 2010. `http://googlewavedev.blogspot.com/2010/03/introducing-robots-api-v2-rise-of.html?utm_source=feedburner&utm_`

medium=feed&utm_campaign=Feed%3A+GoogleWaveDeveloperBlog+%28Google+Wave+Developer+Blog%29 last checked 03/05/2010.

[Dav05] Thomas H Davenport. Thinking for a Living: How to Get Better Performances And Results from Knowledge Workers. *Harvard Business School Press*, 2005.

[Dix97] Alan Dix. Challenges for Cooperative Work on the Web : An Analytical Approach. *Computer Supported Cooperative Work*, 6:135–156, 1997.

[DVS$^+$98] Alan R Dennis, Joseph S Valacich, Cheri Speier, Michael G Morris, and Bookmark. Beyond Media Richness : An Empirical Test of Media Synchronicity Theory I . Introduction II . Literature Review. *Review Literature And Arts Of The Americas*, 1998.

[Eva93] Remy Evard. Collaborative Networked Communication: MUDs as Systems Tools. *Proceedings of the 7th USENIX conference on System administration*, 1993.

[Fer09] Andrés Ferrat. *Google Wave: Up and Running*. O'Reilly Media, 2009. http://oreilly.com/web-development/excerpts/9780596806002/google-wave-intro.html last checked 11/10/2009.

[Fox05] G. Fox. Global multimedia collaboration system. *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems, 2005.*, pages 11–13, 2005.

[Gj03] Robert Godwin-jones. Blogs and Wikis : Environments for On-line Collaboration. *Language Learning and Technology*, 7(2):12–16, 2003.

[Goo09a] Google. About Google Wave, 2009. http://wave.google.com/help/wave/about.html last checked 12/08/2010.

[Goo09b] Google. Google Wave Developer Guide, 2009. http://code.google.com/intl/de/apis/wave/guide.html last checked 11/10/2009.

[Goo09c] Google. Google Wave Robot API, 2009. http://code.google.com/intl/de/apis/wave/extensions/robots/index.html last checked 03/02/2010.

[Goo09d] Google. White Paper: Google Wave Federation Architecture, 2009. http://www.waveprotocol.org/whitepapers/google-wave-architecture last checked 11/10/2009.

[McA06] Andrew P. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration. *MIT Sloan Management Review*, 47, 2006.

[Mih09] Adrian Mihai. Improving Knowledge Sharing in an Open Informal Network : Knowledge Management in Open Coffee. *Dissertations, Dublin Institute of Technology*, 2009.

[RF09] Kai Riemer and Stefanie Filius. Kontextualisierung der Medienwahl mit Hilfe von Kommunikationsgenres. *Wirtschaftsinformatik*, 51(2):192–205, 2009.

[Sie09] MG Siegler. Google Wave Drips With Ambition. A New Communication Platform For A New Web., 2009. http://techcrunch.com/2009/05/28/google-wave-drips-with-ambition-can-it-fulfill-googles-grand-web-vision/ last checked 12/08/2009.

[SLP09] Jason L Snyder and Joo Eng Lee-Partridge. Understanding Choice of Information and Communication Channels in Knowledge Sharing. *Proceedings of the International Conference on Information Systems*, 2009.

[TK94] Gunnar Teege and Michael Koch. Integrating Access and Collaboration for Multimedia Applications. In *Proceedings of International Conference on Multimedia, Hypermedia and Virtual Reality*, pages 170–176, Munich, 1994.