Service-based Development of Mobile Real-time Collaboration Applications for Social Networks

Daniel Schuster, Thomas Springer, Alexander Schill
TU Dresden, Computer Networks Group
Dresden, Germany
{daniel.schuster, thomas.springer, alexander.schill}@tu-dresden.de

***Abstract*—Social Networks will unfold their full potential when connected people are enabled to collaborate - any time, appropriate to the current location, activity and computing environment. However, development of collaborative applications in pervasive environments is still a big challenge.**

In this paper we present a service-oriented approach for the development of collaborative applications on top of social networks. It consists of a conceptual platform providing a set of generic services for real-time collaboration and the integration of existing social networks. In an experimental evaluation we implemented the conceptual platform for Android devices using the Extensible Messaging and Presence Protocol (XMPP) family together with a set of reusable services for Facebook integration, user location and proximity detection, media sharing, and collaborative editing. We developed several case study applications by composing generic collaboration services to demonstrate the feasibility and value of the approach.

***Keywords*-Real-time Collaboration; Mobile Social Networking; Service-oriented Platform; Extensible Messaging and Presence Protocol (XMPP); Android;**

I. Introduction

Social networks have developed into widely-used platforms for interconnecting people with support for collaboration features like chatting or content sharing. The availability of rich features for collaboration - any time, adapted to the users current context - would unfold the full potential of social networks. Especially features for real-time collaboration - the IP-based communication and collaboration with synchronous interaction among geographically distributed participants - with features like shared editing of documents would increase the value of social networks. But even if recent technological innovations enable resource rich mobile devices and constant wireless connections with sufficient data rate, the integration of rich collaboration features across web-based and mobile platform is still a big challenge.

Service developers are confronted with a large number of highly heterogeneous platforms, partitioned into separated islands with respect to user communities, (user-generated) content data, provided services and technological base. As a result, for the creation of value-added services on top of social networks usually either a certain platform has to be selected (either web-based or mobile) or basic social networking functionality like user management, profile handling and collaboration features have to be re-implemented. Advanced functionality for real-time collaboration in wireless network infrastructures like shared editing requires high development effort and the adoption of specific communication protocols. In consequence the development of value-added services in the domain of mobile real-time collaboration (MRTC) is very complex and costly.

In this paper we present a service-oriented approach to improve the development of MRTC applications. With the Mobilis platform [**?**] we already defined a conceptual architecture consisting of four layers and several collaboration services. We now refine this concept through the course of this paper with a mapping to concrete technologies and a client/server architecture. The paper describes three main contributions: First, we define an XMPP-based realization of the **architecture** defined in [?] that builds upon standardized XMPP extensions (XEPs) to support real-time collaboration functionality in a service environment. Broker services act as XMPP clients thus leveraging XMPP as an infrastructure for service user to service provider communication. Second, we provide a **set of reusable collaboration services** for group management, social networks integration, location updates, shared editing of XML documents and media sharing. These services can be composed to ease and accelerate the development of applications. Third, we provide several **case studies** with implementations of different MRTC applications to show the feasibility of our approach.

In the following, we first analyse the requirements for service-based development of MRTC applications in Section II. We present the architecture of our service environment and introduce a set of reusable services for MRTC in Section III. Our case study and discussion of results is described in Section IV. We end the article with a summary and an outlook to future work.

II. Requirements Analysis

To get an impression of the need of the developers of MRTC applications we performed a technology survey in [?] comprising the top 50 proposals of Google's Android Developers Challenge. More than 80% of these applications use at least one collaboration feature while 30% combine at least three collaboration features. Most of these applications

where built from scratch, thus re-implementing collaborative functionality. Some of the applications already provide coupling to a special social network (often Facebook), but many of them rely on their own user basis to be built.

The following **general requirements** to a service platform for MRTC applications can be derived from this observation. They are devided into three functional and three non-functional requirements:

A. Functional requirements

Enriched presence: The provisioning of enriched presence information as well as intuitive means for users to manage preferences is one of the major prerequisites for MRTC applications. This comprises at least the ability to localize other participants by a world-wide unique identifier and to be able to use this identifier to send messages as well as to call them directly. Furthermore, information on the user's availability, device, location, activity and further context are of importance to many MRTC applications.

Collaboration services: An extensible set of collaboration services shall be provided which are commonly used by a number of MRTC applications eliminating the need to re-implement their functionality. This includes basic communication features, media sharing, shared editing and dynamic group formation and management.

Web 2.0 integration: MRTC applications should not only create their own user communities but coexist with and complement the world of Web 2.0 applications like social networks, media sharing portals, location-based services, blogs and forums. The platform should provide connectors to existing Web 2.0 applications, emphasizing support of existing social networks.

B. Non-functional requirements

Support of heterogeneous mobile platforms: As the landscape of mobile devices is very heterogeneous, multi-platform support is essential for any MRTC service environment. It is not sufficient to only stick to one mobile environment like the iPhone, Google Android, or Windows Mobile but to define open protocols that may be equally implemented on a number of platforms. Support of mobile platforms also includes mobility issues like handling disconnections and varying network bandwidth and delays.

Configuration, adaptation, and extensibility: Due to the heterogeneity of mobile devices and wireless network technologies a "one-size-fits-all" application would fail to meet the challenges raised by heterogeneity. The MRTC service environment should thus be configurable and include adaptation mechanisms to be able to work in a number of different environments. Furthermore, the ability to add and modify services of the platform (extensibility) is of great importance regarding the fast development in the area of MRTC applications in the last years.

Security and privacy: Special security requirements arise from the new possibilities of the platform. If I am able to publish my location as well as to use information from my social networks, this information should be handled with care. A compromise between enriched presence features and privacy has to be found. The service environment should therefore support the user in controlling who sees what information and to enforce fine-grained access policies.

As a secondary non-functional requirement, we also want to mention **compliance to open standards** here. There are already a number of open standards in the area of communication and collaboration such as SIP, XMPP and the emerging Google Wave. Standards-compliance can already be derived from the requirements multi-platform support, Web 2.0 integration, and extensibility, but it also has a further dimension by its own. Any new collaboration platform not compliant to open standards would only create a new island of isolated users not able to communicate with anyone outside this island. Thus standards-compliance is essential for any MRTC application.

In the following, we want to show how these requirements can be fulfilled using open standards as a starting point. We first introduce the architecture of our service environment before discussing concrete MRTC applications building upon the platform.

III. The Mobilis service environment

After discussing several design alternatives in detail, we decided for the eXtensible Messaging and Presence Protocol (XMPP) [**?**] as the basis for our service environment. Classic approaches for reusable services like SOAP or REST are not adequate for MRTC with its real-time requirements. XMPP already contains a lot of useful extensions called XMPP Enhancement Proposals (XEP). Many basic services needed for collaboration such as multi-user chat, service discovery, file transfer, and enriched presence are already part of XMPP and its XEPs. Furthermore, XMPP provides a good extension mechanism making it easy to integrate own services into the architecture.

To design a highly flexible architecture, instead of extending a special XMPP server implementation we use an XMPP server as a black box as illustrated in Figure 1. The services of the platform are provided as XMPP resources with their own ID thus acting like ordinary XMPP clients. XMPP offers the possibility to identify each potential collaboration participant by a unique ID in the form *user@server.org/resource* where each user may be logged in with more than one resource at the same time (e.g., PC, notebook, iPhone). This mechanism is the core of our service environment as it enables the clients to connect to the different services of the service environment. A permanent connection is established between client and collaboration service allowing for fast exchange of real-time data, e.g., for shared editing.

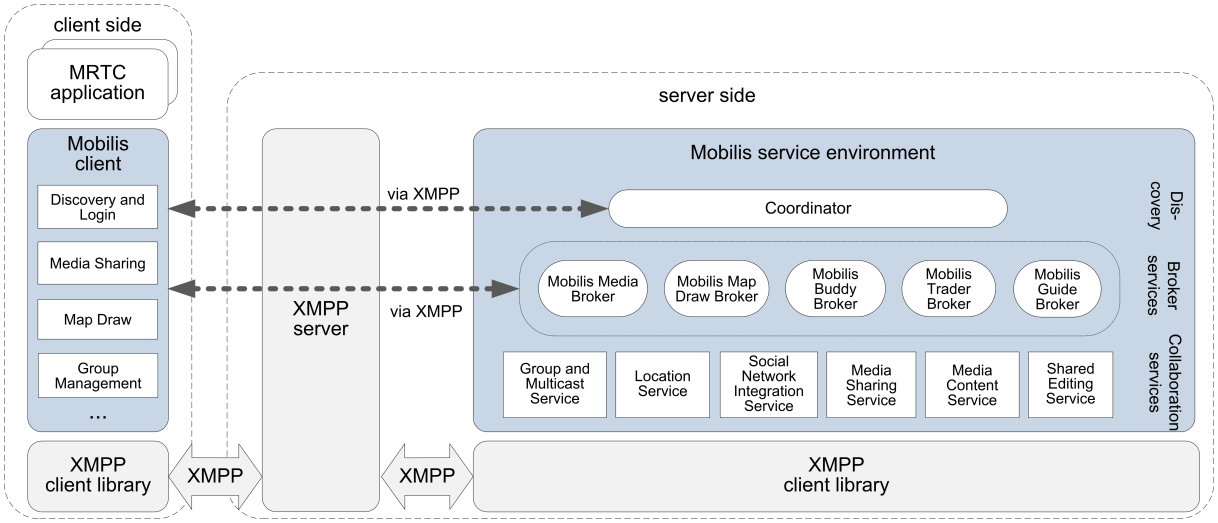

Figure 1. Architecture of the Mobilis service environment

Each Mobilis service environment contains exactly one Coordinator which also acts like an XMPP client resource and connects to the XMPP server using a well-known ID such as *mobilis@server.org/coordinator*. We use the service discovery extension XEP-0030 [**?**] to find so-called Broker services. A Broker service bundles functionality used by one or more MRTC applications. These Broker services again act like XMPP client resources allowing to directly connect Mobilis clients to their respective Broker service. The Broker services register themselves at the Coordinator to be found by the discovery.

One layer below the Broker services a number of collaboration services exists offering the actual **reusable functionality** of the platform. Each Collaboration service may be used by one or more Broker services. Thus reuse of functionality is possible within the server without creating too much overhead as the collaboration services and their features are not included in the service discovery.

The functionality and mode of operation of the collaboration services is described in the following.

A. Group and Multicast Service

If a user wants to join a collaboration group, a request is issued to the local instance of the group management service. It contacts the Coordinator with an info/query XMPP message with type *get* and the namespace *mobilis:iq:groups* returning a list of items with the name of the each group and their respective IDs. The user then issues a *JoinGroup* request to the Coordinator using again info/query XMPP messages. Once the group is established, a multicast mechanism is available to send messages to the whole group. These may be simple chat messages or complex protocol messages such as difference updates for shared editing.

B. Location Service

The location service receives location updates from the repspective clients and forwards them to interested services in a publish/subscribe manner. Some of the Broker services only need location updates for proximity checks such as the Mobilis Buddy broker who gives an alert whenever one of your Facebook contacts using the same application is in your proximity. While a user may be willing to share his position for proximity checks, he may want to restrict direct access to his location. So the location service is not accessible directly from the clients, but mediated by the Broker services enabling application-dependent access restrictions.

C. Social Network Integration Service

The Social Network Integration Service is targeted at the requirement Web 2.0 integration. It provides an abstract interface to login to a social network and to get a list of friends with their respective XMPP IDs. Often this type of access requires the registration of an extension application at the social networking platform, e.g., a Facebook app for the Facebook platform. In the case of Facebook, an authentication token is acquired which has to be authorized by the user. This token can then be used to retrieve the list of Facebook friends. The mapping to internal IDs is done at the Social Network Integration Service. MRTC applications building upon the Mobilis service environment can thus make use of the community structures already available.

D. Media Sharing and Content Services

The Media Sharing Service and the Media Content Service allow to register and unregister media objects like pictures or videos tagged with meta data about geo-location, owner and time as well as self-defined tags. It is then possible to browse the media collection using again XMPP

info/query messages defined in an XMPP extension protocol on media sharing. The actual transfer of media objects is already possible with XMPP means using the standardised XMPP extension SI File Transfer [**?**]. We devide media sharing functionality in accessing the repository and meta data (Media Sharing Service) and storage and retrieval of media files (Media Content Service) as this enables the distribution of these services for better scalability.

E. Shared Editing Service

As one of the key components for sharing applications, we offer a service for shared editing of XML documents. It builds upon the Collaborative Editing Framework for XML (CEFX) [**?**] allowing for concurrent editing of XML structures in real-time using operational transformation algorithms. It is a very powerful and flexible service as these XML structures may be formatted text documents as well as graphics files (SVG) or geo-information (KML). A local copy of the XML file being edited in the session is maintained at each client to ensure timely updates in the case of edit operations. The changes are then sent with a message *propagateLocalOperation* to the server who then applies the changes to the server copy, resolves any conflicts and sends back changes to be made to have consistent versions of the document on all clients (*handleRemoteOperation*).

IV. Case Studies

To evaluate the feasibility of our approach, we implemented several case studies on top of the platform using the Android developer SDK V1.5, G1 phones, and the OpenFire XMPP server. In the following, we introduce MobilisMedia, MobilisMapDraw, MobilisGuide, MobilisBuddy and MobilisTrader to demonstrate the reusability of the services defined in Section III. Moreover, we discuss the experiences made with our approach, especially with respect to XMPP, Android and service composition.

A. MobilisMedia

MobilisMedia allows the user to share image and video files that were created using the mobile phone at a certain location and time. The left part of Figure 2 shows a detail view allowing to download or replace an image in the media repository. The repository itself consists of a multi-dimensional meta data store and a content store.

Location information can be used to create a map-based view of a media repository. In addition, the user may also look for images specifying a specific user and/or time span when the image was created. The protocol of the *Media Sharing Service* described in the last Section is generic enough to support such SQL-like queries and returns a list of media files to the client. The transfer to the *Media Content Service* is done using XMPP mechanisms and thus even works between Mobilis Media and any other Jabber client that supports file transfer.

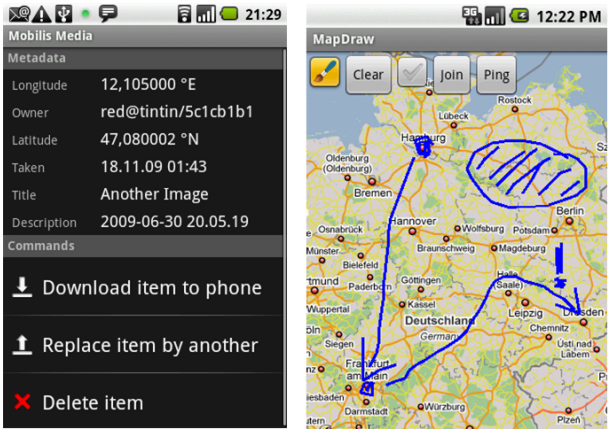

Figure 2. Screenshots of the Mobilis Media (left) and Mobilis Map Draw (right) case studies

Using the *Social Network Integration Service*, images can be shared between people connected in a social network. The user can take pictures while on travel and submit them to the media repository restricting access to all his Facebook friends. The *Location Service* is used to set up a closed group among these participants. It is even possible to keep the files on the Android phone and to only publish the meta data to the repository. File transfers requested by remote participants may then be accepted or rejected individually by the owner thus keeping his images under control.

B. Mobilis Map Draw

Mobilis Map Draw enables users to collaboratively draw on a map. The drawing operations are propagated in real-time to ensure a consistent view shared by all participating users. The right part of Figure 2 shows the current state of the collaborative session where some friends connected via Facebook discuss a common trip from Hamburg to Dresden. It is visualized as an overlay of a Google Map. The underlying *Shared Editing Service* supports editing of arbitrary XML content. For the Mobilis Map Draw the edited content is a KML file which allows to store drawings with geo-coordinates. In this way all session participants share the same content, even if the individual map visualization is different. The drawings are correctly placed and scaled to the current map section. Moreover, drawings and routes can be exchanged with external tools like Google Earth.

As already described for Mobilis Media, Mobilis Map Draw uses the *Social Network Integration Service* and the *Group Management Service* to set up closed groups for collaborative editing on a map based optionally based on contacts from social networks.

C. MobilisGuide

MobilisGuide is collaborative tourist guide where members of a closed group can see each other on a map and discover a city. The application provides a map-based view

enabling extended presence of the grouped tourists. The *Group and Multicast service* is the basis to maintain closed groups and the communication among group members via multi-user chat. The *Location* service enables the sharing of location information within the group. In addition, the *Media Sharing* service allows tourists to publish and access multimedia content like photos or short video clips with other group members.

D. MobilisBuddy and MobilisTrader

MobilisBuddy gives an alert if one of your Facebook friends is in your close proximity and allows to contact him or her. MobilisTrader is a location-based trading application allowing to search for specific goods in your proximity. Both applications set up on Facebook by the means of the *Social Network Integration* service and use the *Location* service to realize proximity-based mechanism and the *Group and Multicast service* for multi-user chat.

E. Discussion

We believe that the full potential of social networking can be exploited only in combination with collaboration features and vice versa. We have demonstrated the value of a service-based approach which allows the creation of MRTC applications by composing **collaborative services**. Our case studies illustrate the reusability of collaboration services in different applications. This avoids the re-implementation of functionality for shared editing, media sharing, dynamic grouping and location management from scratch and thus, increases the efficiency of application development. Particularly important is the **integration of existing web-based platforms** which has been demonstrated with the provisioning of the Social Network Integration Service implemented for Facebook. While it enables the integration of existing user communities further work is required to also integrate content data and platform functionality to overcome the isolation of mobile and web-based social networks.

XMPP has proven to be an **open standard** appropriate for the implementation of real-time collaborative applications in social networks. All described functionality and services could either be mapped directly to the XMPP protocol and its extensions, or be implemented using XMPP-compliant own extension protocols. The fundamental extensibility of XMPP and its openness for various interaction paradigms like request/response and publish/subscribe ensures its adoption in many further application areas. The concept of representing all Mobilis clients and Broker services as XMPP clients in combination with a standard XMPP server infrastructure ensures high **flexibility, scalability and extensibility** of the approach. Each part of the environment is loosely coupled so that new Mobilis components can be added easily. The XMPP server technology is exchangeable.

Anyway, XMPP is not particularly designed for wireless networking infrastructures and mobile devices. Especially, XMPP sessions depend on persisting TCP connections. This requires at the one hand a constant connection which is not guaranteed in wireless networks. At the other hand the energy consumption is might be higher. Moreover, in the case of network outages, established sessions are immediately terminated. In consequence, the login procedure and session setup have to be repeated for connection re-establishment. A possible solution with respect to network outages might be XEP-0198 Stream management which contains a mechanism for stream resumption.

Our current implementation is based on Android which supported the development of our case study applications in many respects. Especially, the map access and views, gathering of location information from GPS and distance measuring are well supported by standard functionality provided by Android. In addition the general concepts of Android for application and user interface development, namely intents, activities, and services enabled a fast implementation of the Mobilis service environment.

Nevertheless, the goal of our approach is **multi-platform support**. Based on the platform independent specification we will implement the Mobilis service environment on further platforms in the future, namely iPhone and Windows Mobile. Because the protocols all rely on XMPP and are defined as XMPP extensions, they may be reimplemented easily on any platform that supports XMPP. Our strategy of adopting platform specific technologies has proven to be successful, especially with respect to user interfaces [**?**].

V. Related work

Proem [**?**], ContextPhone [**?**], and the George Square System [**?**] are examples of frameworks focusing on different aspects of collaboration. Proem includes services realized as a high-level Java API that manage areas like presence, community management and a common data space. ContextPhone enables mobile applications to use sophisticated context information on top of the Nokia Series 60 platform. The George Square System is a map based collaborative system for tourism. It primarily analyses how tourists work together in groups and collaborate around maps. However, the underlying communication protocols are application specific and directly define the syntax of messages, thus, no open standards are used.

Google Wave [**?**] is an emerging Web application for instant communication and collaboration supporting shared editing of XML documents, instant sharing of pictures, and integration of third-party gadgets through an extension API. It is based on open protocols which allow interoperation between different Wave servers. Especially, Google Wave Federation protocol is based on XMPP. It includes the possibility to merge updates for concurrent XML editing and to ensure consistency on all copies of the wave.

Regarding these existing environments for MRTC applications, many of them are platform-specific, restricting the use,

e.g., only to Nokia phones. Collaboration services offered by these frameworks often only include basic communication and context gathering features. Especially shared editing of XML documents and media sharing is often not part of the frameworks. These features are fully covered by Google Wave, so why is there a need for yet another platform? As Google Wave is completely Web-based, the individual features of Google Wave can not be embedded as services in own applications developed for mobile phones. Services and Web applications can be integrated into Google Wave but not vice versa. So we believe that service environments targeting the needs of developers of MRTC applications will not extinct because of Google Wave. They will rather coexist with this new form of mixed-mode communication and provide appropriate interfaces. This is especially easy to realize within the Mobilis environment, as the Google Wave Federation protocol is XMPP-based and thus our Mobilis server could also act as a proxy using XMPP messages to synchronize with Google Wave servers.

Regarding the area of mobile social networking, SAMOA [**?**] is an example for mobile social networks who create groups of users based on physical proximity, common affinities, attitudes, and social interests. As a companion approach, the MobiSoC [**?**] middleware for Mobile Social Computing provides an API to develop new mobile social networking applications that can use generic services for finding users of the same platform in close proximity of other mobile-specific functionality. The main drawback of these and other comparable approaches is that they create a new social network instead of using the profiles and person links already available in existing networks.

While integration of existing social networks in MRTC applications is already possible by means of APIs like the Facebook API or the OpenSocial API, this requires an extension application for each of the MRTC applications to be written for each of the social networks the MRTC app wants to support. Our environment only needs one such extension application per social network for the whole platform thus drastically reducing the effort for social network integration.

VI. Conclusion

Based on our observation of applications developed for mobile phones, there is a strong need for a service environment supporting the collaboration functionality and integration of existing Web 2.0 applications. We developed such a service environment based on open standards, especially XMPP and its extensions. As can be seen from this paper, XMPP at its current mature state is more than appropriate to serve as the core of such an environment. We used existing extensions for service discovery, group management, and file transfer and added new extensions for location updates, shared XML editing, media repository management, and social network integration. With a number of showcase applications built on the Android platform the flexible composition of MRTC applications based on reusable services has been demonstrated.

In the future we plan to cover more target platforms such as the iPhone and Windows Mobile. We plan to make the source code of our platform publicly available to foster the contribution of other interested parties to the platform as well as the use in future mobile applications.

Acknowledgement

The authors would like to thank István Koren, Benjamin Söllner, and Dirk Hering for their contributions to the Mobilis service environment and MobilisMedia and MobilisMapDraw case study applications. The work of the authors was partially supported by the German Federal Ministry of Education and Research (BMBF) under grant BRA 06/037.

References

[1] Xmpp standards foundation, 2008. http://xmpp.org.

[2] D. Bottazzi, R. Montanari, and A. Toninelli. Context-aware middleware for anytime, anywhere social networks. *IEEE Intelligent Systems*, 22(5):23–32, 2007.

[3] B. Brown and E. Laurier. Designing electronic maps: an ethnographic approach. In *Map Design for Mobile Applications*. Springer Verlag, 2004.

[4] A. Gerlicher. *Developing Collaborative XML Editing Systems*. Phd thesis, University of the Arts London, 2007.

[5] Google. Google wave - communicate and collaborate in real-time, 2009.

[6] A. Gupta, A. Kalra, D. Boston, and C. Borcea. MobiSoC: a middleware for mobile social computing applications. *Mob. Netw. Appl.*, 14(1):35–52, 2009.

[7] J. Hildebrand, P. Millard, R. Eatmon, and P. Saint-Andre. Xep-0030: Service discovery. Technical report, XMPP Standards Foundation, 2008.

[8] G. Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):62–64, 2002.

[9] T. Lange, M. Kowalkiewicz, T. Springer, and T. Raub. Overcoming challenges in delivering services to social networks in location centric scenarios. In *LBSN '09: Proceedings of the 2009 International Workshop on Location Based Social Networks*, pages 92–95. ACM, 2009.

[10] T. Muldowney, M. Miller, and R. Eatmon. XEP-0096: Si file transfer, 2004.

[11] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005.

[12] T. Springer, D. Schuster, I. Braun, J. Janeiro, M. Endler, and A. A. F. Loureiro. A flexible architecture for mobile collaboration services. In *Companion '08: Proceedings of the ACM/IFIP/USENIX Middleware '08 Conference Companion*, pages 118–120, New York, NY, USA, 2008. ACM.