HiveMind: Cross-platform Application for Collaborative Mind Mapping

Andrey Vasilev, Andrey Golovchenko, Alexander Kulikov, Ilya Paramonov
Demidov Yaroslavl State University
Yaroslavl, Russia
Email: {vamonster, golovchenkoaa, ivparamonov}@gmail.com

Abstract

Mind mapping is a process of data representation in the form of hierarchical structures, named mind maps. This process often involves many people. In this situation mobile devices open new capabilities in collaboration. Unfortunately existing tools for mobile devices have a lack of functionality, existing on-line services usually require paid accounts or do not take into consideration peculiarities of mobile devices. That is why we decided to start the project HiveMind targeted at development of collaborative mind mapping software.

In the article we present our view of collaborative mind mapping involving mobile devices, describe current functionality and architecture of HiveMind application and discuss choosing middleware for collaboration implementation.

I. Introduction

Mind map is a means of hierarchical data representation which is often used in study, project management, problem solving, brainstorming and many other activities.

Mind mapping often involves many people of different occupations. It is not easy to provide all the members of a discussion with an equal ability to contribute. It is a real challenge to collaborate when participants are spread across space. The lack of approaches for distributed collaboration leads to loss of productivity and someone's ideas.

Mobile devices open new capabilities in collaboration. They would overcome the mentioned problems. When using such an approach no one gets left behind in discussion despite his or her location.

There are some mind mapping applications available for mobile platforms: MindNode (touch) [1] and MindJet [2] for iPhone, MindMap Memo [3] for Android, Labyrinth [4] for Maemo. Unfortunately these applications have scanty functionality and no support for collaborative work. MeisterLabs [5] offers a paid on-line service MindMeister for collaborative mind mapping and an application for iPhone interacting with this service. Mind maps created by the user are stored in the MindMeister storage and cannot be edited off-line.

Due to these shortcomings we decided to start the project HiveMind. There are two main goals of this project. The first one is to create a concept of collaboration involving mind maps and mobile devices. The second goal is to develop an Open Source cross-platform application (also called HiveMind) which allows collaborative mind mapping.

II. Use Cases For Collaborative Mind Mapping

In this chapter we present our view of collaborative work involving mind mapping. There are some possible scenarios below.

A user creates a mind map and starts editing it using his/her mobile device, laptop or personal computer. During mind mapping he/she decides to show his/her work to a friend (to

get some help, for example). To do this, he/she publishes the mind map as a network service and goes on editing. In this case the user's device or laptop works as a server available from a local network or Internet.

Another user connects to the network service and retrieves the latest copy of the published mind map. After that both users can edit the map together. The number of possible participants is potentially unlimited, so other users can also join collaboration. All the changes are sent immediately to all participants of collaboration.

To simplify discussion of the work for all participants, mind mapping application may provide a text messaging service. It would allow users to exchange suggestions, questions and opinions without necessity of using additional applications.

Another use case of collaborative mind mapping relates to public speaking or brainstorming. For example, the speaker provides additional material to the participants, such as a plan of the presentation, detailed speech synopsis, links to the additional resources. The other participants can see these materials using their mobile devices/laptops. They also receive all the changes that the author makes during his/her speech, explaining details on-the-fly.

The mind map may contain some materials for a live discussion. In this case it can be used as a shared flip chart with the hierarchical structure for gathering participants' ideas, comments and opinions. The advantage here is that the results become available immediately.

At last, if the author would not like participants to edit the published materials, he or she can prevent any modifications by using read-only mode. Read-only mode can be also applied to a part of the mind map only.

III. Implementation Of Standalone Mind Map Editor

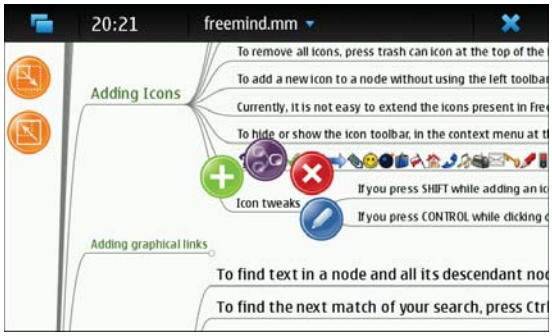
We started software implementation from developing a standalone mind map editor. We have chosen the following development tools: Python programming language, Qt Framework (as it is officially supported on Maemo), PySide bindings library which allows to use Qt classes from Python code, and Eclipse IDE with PyDev plug-in.

FreeMind [6] is the most popular Open Source mind map editor for PC. Its file format is supported by the majority of mind map editors. That is why we decided to use FreeMind file format for input/output for the sake of interoperability. So mind maps created on mobile device can also be opened with FreeMind or another mind mapping software on PC and vice versa.

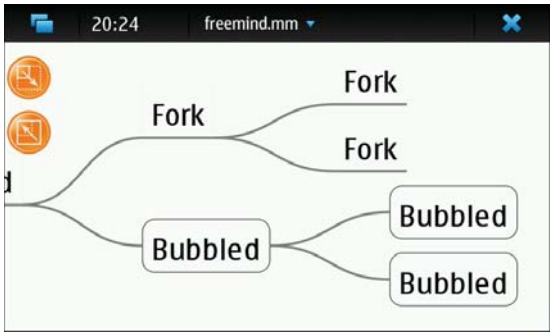
To provide full support for FreeMind file format we investigated all available information about it. Information about some unclear or undocumented details were elicited directly from FreeMind source code. The obtained data were compiled into a single document containing description of all the elements and attributes valid for this XML-based format. To ensure robust parsing of mind maps created by means of FreeMind we developed a unit test suite.

For now, HiveMind has the following features:

- Reading/writing mind maps in FreeMind format;
- Basic node operations: addition, removal, transfer;
- Alteration of node text and formatting (background color, text color, size, style and family);
- Icons selection for a particular node;
- Node subtree folding;
- Inheritance of formatting for attributes which were not set explicitly;
- Edge shapes (linear, bezier, etc) and labels (not supported in FreeMind);
- Undo/redo capability.



a) mind map view and context menu



b) zoom-in and node bounding

c) icon selector

d) edge editor

Fig. 1. HiveMind screenshots

Several screenshots showing the mentioned functionality are demonstrated in fig. 1

Initially we planned to develop a new mind mapping application for Maemo platform only. However effective collaborative work seems to be incomplete without an ability for desktop/laptop users to participate. These users may use FreeMind, but it does not support collaborative work. There is an ability of extending FreeMind functionality by plug-ins, so this approach can be used to implement interaction between FreeMind and HiveMind. An alternative approach is to develop a cross-platform application suitable for both mobile and PC platforms. Developing HiveMind as a cross-platform application has some advantages, as it is shown below.

FreeMind has been being developed for ten years and evolves slowly nowadays. Its source code contains plenty of long functions (50 lines and more), which makes this project rather difficult to contribute. So it would be probably easier to create a new application instead of extending the existing one. Another advantage relates to a possibility of adding new features having no analogs in FreeMind, such as edge labels or specific attribute inheritance.

Because of the use of Qt Framework, it is possible to run the application on various platforms without any modification. Nonetheless it is better to make some particular interface adjustments in order to accomplish seamless interface on every supported platform. For example, node editor dialog is split into the two parts on Maemo (fig. 2) due to the lack of space on the tablet screen.

Meanwhile, there is no need of the separate font properties dialog on PC, so it is possible to place all the widgets in the single dialog (fig. 3).

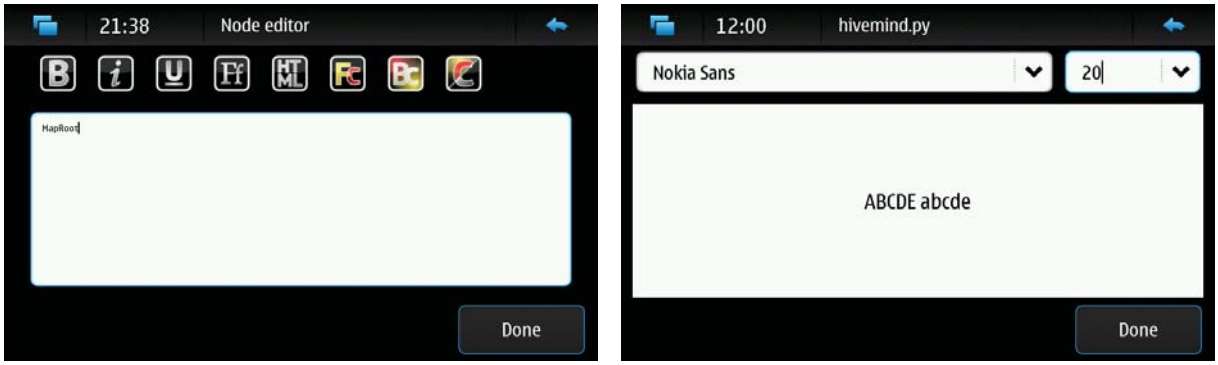


Fig. 2. Node editor dialogs on Maemo

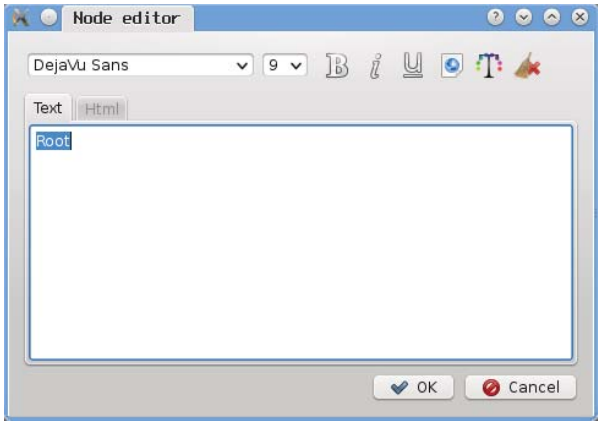


Fig. 3. Node editor dialog on PC (GNU/Linux)

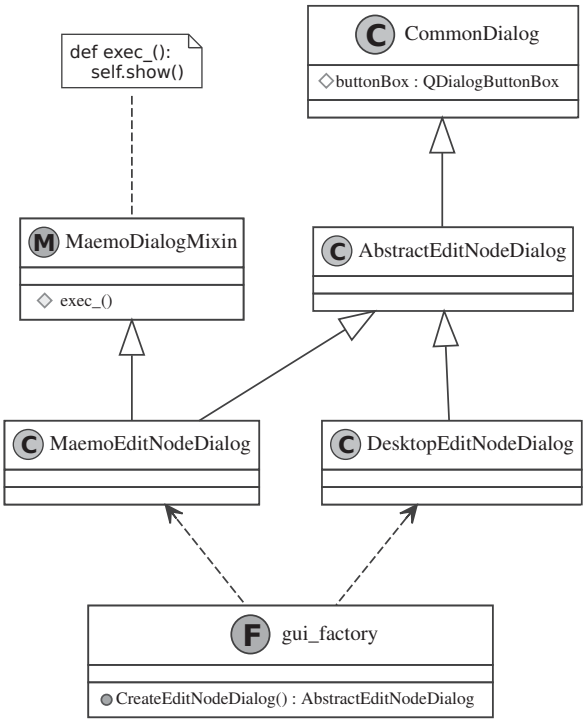


Fig. 4. Hierarchy of node editor classes

In order to take such differences into account we modified the architecture of HiveMind and utilized the factory pattern to instantiate appropriate dialog classes for each specific platform. For instance, the class diagram related to the node editor dialog hierarchy is shown in fig. 4.

CommonDialog is a superclass of all dialog classes. It contains an instance of QDialog-ButtonBox class, which provides common dialog buttons for the current platform.

AbstractEditNodeDialog is an abstract class, containing common widgets and platform-independent code of node editor dialog. The platform-dependent code (in the first turn, it is the widget layout code) is located in descendants of AbstractEditNodeDialog: DesktopEditN-odeDialog and MaemoEditNodeDialog classes.

MaemoDialogMixin is a class inherited by all Maemo dialogs. It overrides exec_() method of QDialog class and provides specific dialog initialization for Maemo stacked windows support.

Gui_factory is a Python package being a detector of the current platform of running. It provides the factory functions, which instantiate appropriate dialog classes corresponding to the current platform.

IV. Middleware For Collaboration Implementation

There is a lot of ways to achieve network communication between applications starting from assembling raw TCP/IP datagrams up to using object-oriented libraries for specific application-level protocols. We stated a list of requirements to choose an appropriate middleware technology. These requirements were collected from the three sources: our view of collaboration, restrictions imposed by the target platforms and a preliminary architecture of the application.

According to our view of collaboration (see section II) we have a server and several clients. For the sake of simplicity we assume that a server is allowed to be created anytime. Clients connect to the server and modify the shared mind map. Every change of the map is propagated via the server, which notifies all the clients about it. For robust communicating all the participants must always have the exact copy of the mind map.

HiveMind is now targeted at both mobile devices and PCs. Each platform gives us unique advantages. PC users usually have wide broadband Internet connection. Mobile devices can gain Internet access almost anytime and anywhere. Due to the lack of IPv4 addresses mobile operators often assign private IP addresses to mobile devices that makes them unreachable from the Internet. Data transmission via wireless media sometimes causes unreasonable delays for several seconds or more. PC users can use NAT or complex firewall software that blocks incoming connections. It may be even worse: in corporate environments users are often limited to HTTP proxy as the only source of the Internet connection.

Summarizing all these challenges we get the following list of requirements for middleware technology.

- Communication is based on the client-server model.
- Server supports for subscription.
- Users may have a slow unreliable connection to the Internet through the NAT gateway.
- There must be no additional effort to set up the server.
- Middleware must be supported by a Python library.

We compared several alternative technologies and chose the XMPP protocol as it allows us to avoid almost all concerns about implementation of client-server communication.

XMPP protocol can be considered at the several levels of abstraction. At the low level there is a XMPP client application that connects to the XMPP server and exchanges messages with it. Connection can be established in many ways, even on top of HTTP protocol. At the higher level of abstraction the client gains access to the whole network, where every node has unique identifier (JID) and no additional effort required to exchange information between them. The client creates a message addressed to another client and sends it to the server, which handles all the routine to deliver the message.

Another great feature of XMPP is protocol extensions (XEP). XMPP is an open protocol and anyone may create a custom high-level protocol on top of it. XMPP Standards Foundation manages the process of creation and maintenance of such protocols. There is XEP-0060 Publish-Subscribe extension [7]. It defines how to implement publish/subscribe services on top of XMPP. Any member of the network can create pub/sub service. XMPP protocol with Publish-Subscribe extension fits the best into our requirements and that is why it is our choice for middleware technology.

XMPP protocol is XML-based, and it may seem that using this protocol would require broadband Internet connection, but it is not so. Intercommunication between XMPP client and server may be compressed when XEP-0138 (Stream Compression extension) enabled.

Nowadays a lot of public XMPP servers (e.g. jabber.ru) have support for chatting. And it is well-known, that participating in chats is successful even when using mobile phones without broadband Internet connection. From the technical point of view collaborative mind mapping looks very similar to chat participation and expected traffic volume looks comparable. The only exception relates to the device being a server, which may require broader connection to communicate with lots of clients. This question will be addressed during the further development of the HiveMind network subsystem.

The next task was to find a pure Python XMPP library that supports XEP-0060 extension. There are eight Python libraries listed at the XMPP Foundation Website [8]. Some of them are discontinued (jabber.py), some are targeted at newer (2.6, 3.x) Python releases which are unavailable on Maemo platform (pyxmpp), the others represent research projects and are not suitable for production environment (SleekXMPP). Twisted is the only XMPP library available in Maemo repository.

Twisted framework does not include support for Publish-Subscribe extension but there is an external full-featured implementation named wokkel.

V. Conclusion

For now we have a cross-platform mind map editing application with a draft implementation of collaboration functions. It is possible to send the whole mind map to the user on connection and to send mind map changes via XMPP protocol by means of wokkel library.

The future work relates to extending application functionality as well as determining new use cases for collaborative mind mapping and adjusting HiveMind to support new scenarios.

Current information about the HiveMind project is available at:
https://linuxlab.corp7.uniyar.ac.ru/projects/hivemind.

Acknowledgments

We would like to thank Sergey Balandin for his useful advice, especially for suggestion of developing HiveMind as a cross-platform application that opened new horizons for our project.

References

[1] MindNode (touch). http://www.mindnode.com/mindnode/touch/
[2] Personal Productivity and Collaboration Solutions that Visually Connect Ideas, Information and People—Mindjet. http://www.mindjet.com/
[3] Mind Map Memo. http://www.takahicorp.com/android/mindmapmemo.html
[4] Garage: Labyrinth: Project info. https://garage.maemo.org/projects/labyrinth
[5] Online Mind Mapping and Brainstorming—MindMeister. http://www.mindmeister.com/
[6] FreeMind—free mind mapping software. http://freemind.sourceforge.net
[7] P. Millard, P. Saint-Andre, R. Meijer. *XEP-0060: Publish-Subscribe specification.* http://xmpp.org/extensions/xep-0060.html
[8] The XMPP Standards Foundation. http://xmpp.org