OPAALS PROJECT

Contract n° FP6-034824

WP5: Integration with the Digital Ecosystem Platform

D5.1 - Open Source Software Engineering Process Methods and Tools

Project funded by the European Community under the “Information Society Technology” Programme

Contract Number: FP6-034824

Project Acronym: OPAALS

Deliverable N°: Open Source Software Engineering Process Methods and Tools

Due date: Month 9. 25/04/2007

Delivery Date: April, 2007

Short Description: The paper describes the set of tools implemented in the OKS, as well as their characteristics (pros and cons) in the context of this project. The second section of the deliverable will be about the development process following the distribution of these tools.

Author: Techideas – Javier Noguera

Partners contributed: Techideas

Made available to: OPAALS Consortium and European Community

Versioning

Version	Date	Name, organization
1.0	April	Javier Noguera, Techideas
1.1	May	Javier Noguera, Techideas

Quality check

Internal Reviewers:

1st Internal Reviewer: Thomas Kurz, Technical University of Salzburg

2nd Internal Reviewer:

Dependences:

Work Packages	Indicate which WP could benefit from this Deliverables and if possible with planned tasks, including a clear justification
Partners	indicate which partner could benefit from the reading of this deliverable
Domains	specify the scientific domains involved in this deliverables and clarify to which extend they are covered by this report
Targets	specify which is the target of this deliverable (Domain researchers, public administrations, SMEs, Scientific communities on specific domains, etc.)

Open Source Software Engineering Process Methods and Tools

Table of Contents

Introduction

The process of the software and community development involves the creation and installation of several tools in order to allow the whole community to speak to each other, store documents and information, define processes, etc [5, 11, 12]. This is, to collaborate.

These tools are usually intended to work in a centralized server in the Internet. This centralized approach carries on with all the well-know issues of centralized services (hosting, bandwidth, storage, administration, etc.) plus an extra issue: aggregation of projects.

If someone wants to create a Software Factory, allowing the aggregation of many different projects in a easy and non intrusive way, we have to integrate tools that require no administration or hosting. That means, moving forward to a decentralized world. Decentralization will be achieved in several steps, starting from a classic centralized way to the fully distribution of services. The distributed state will be achieved hand by hand along with the development of the P2P parading and transactions (WP3)

As a part of the Opaals project, TechIDEAS has set up a set of tools commonly used in software development in addition to another set of tools that have been widely used in ongoing communities [2]. In all cases, these tools use standard and open protocols, which makes that many different applications implement can be used as clients.

We will describe the tools in the first part of the deliverable, including the the most relevant benefits provided by such tools and, in some cases, by the community creation. We must agree on that the software development is not just a matter of writing code. This task is also related to the communication among individuals and the emergence of leaders within the project accounting for the capability to organize work, follow discussions, etc. The communication process, in terms of help for newbies, is also important and can be a key point in the success or failure of a project.

The second section of the deliverable will be about the development process following the distribution of these tools. In all cases we do not want to create a new tool/application, but introducing wrappers or small modifications that allow users, in first instance, to work in a decentralized environment and, if possible, to provide as many features as the original tools as possible. In some cases the basic functionality will be good enough for our proposes of decentralization.

Open Source Tools

By its own nature, Open Source Software projects involve different people, from different places around the world, that, in most of the cases cannot work together at the same place, nor at the same time. This alternation nature promotes teams towards the use of collaborative tools that play the role of a face to face conversation among the components of small work groups.

All tools presented in this section are currently used in some Open Source projects and ongoing communities. Teams are using these tools because they have been proved to be useful for working in such environments, combining on-line/off-line behavior. Some of these tools, like the source code repository, have been used for a long time. Some others, like the Wiki are more recent, but the community is quickly adopting them. Nevertheless, most of the tools used in development tasks follow a centralized design. This means that all of these tools have to be installed in a central server where users connect to. As we can easily state, this nature is not scalable or easily administrable when we want to integrate different projects. Thus, throughout the project, tools will be progressively moved from a centralized behaviour to a decentralized one.

These tools, clients and servers, are available on a large number of platforms at zero cost. They have been also written in different programming languages and for different operative systems. They have been developed by the community as the resulting effort of many volunteers.

Source Code Repository

Probably the most widely used tool in software development (commercial and Open Source) is the source code repository, a place where programmers or any other component of the team can upload source code, documents, configuration files, or any other resource file related with the project.

In Open Source Projects only a small part of the users have rights for creating, modify or delete information. This is the core development team who has the top vision of the project and knows which goals have to be achieved. Other people (anonymous) can download and modify files locally, but they cannot modify them in the remote repository. Users may become part of the core team by achieving some milestones. First milestone is usually the availability of solving *bugs* (errors in the application) reported by the core team or the users. Newbies can then achieve more difficult milestones as developing parts of the application being able to create new files modify the application.

The source code repository plays three roles at the same time:

- First of all, it is the remote place where users can save their documents, acting as a backup system. Once the code is in the repository, programmers can delete their own content if necessary, because they can download it again from the repository at any moment.
- The repository is also the share location where users can view all documents and change them. In this case, one of the members of the working group can introduce some modifications in the morning, while other member of the group can continue the same work in the afternoon.
- One of the most relevant features of many common repositories, like the Control Version System (CVS), or the Subversion (SVN), is that they can follow the track of all the changes made in the repository (user who introduced the changes, content that was changed, dates, etc.). This feature is combined with the possibility of marking an retrieving specific versions of the code.
- If there is a conflict with the versions of a file, e.g., when a user wants to upload a new version that another user has already uploaded before, repositories as CVS and SVN try to merge

changes automatically. If the merging of documents is not possible, they allow the user the possibility of editing the content and decide which is the correct version to store. In any case, changes made by both users are not lost, but stored in the repository.

The protocol used for creating, updating, retrieving or deleting documents is well known and, in more recent tools like subversion, uses the standard HTTP/HTTPS.

Wiki

A *wiki* is a web application that shows information (pages written by users) in an HTML form. Users can create, modify or delete content using their web browser, without the need of any other special editor. *Wiki* pages are plain text pages that are converted to HTML using an internal engine transparent to the user. This means that the user does not need to have specific programming skills (i.e., knowledge about HTML, CSS, XSLT, etc.) in order to add or modify the content. Most of the *wiki* applications has a build-in search engine.

The content of a *wiki* is called *wiki-page* and it is usually a definition or a description of a process. The user can read the wiki-page as any other page in the Internet, using his browser. Some of the words within the definition are marked with a link, this means the user can *click* on these links and he will be redirected to other *wiki-page* (definition) where that word is defined. Wiki-pages are highly connected among them. At any moment the user can **edit** the page, adding new content, or deleting content he feels incorrect. At the same time, users can choose to **create** new definitions, as new wiki-pages. Definition is now part of the *wiki,* and can be linked from other wiki-pages, as well.

Due the fact that any page can be modified by other users*, Wiki* engines are an excellent way to create content in a collaborative way.. This means that the content can be easily maintained, and always kept up to date and free of errors, because any user can add the last information, or modify incorrect references.

When editing, the user does not need to write the content in HTML (the content known and interpreted by the browser), but simply plain text. If he wants to write a word in bold, for example, he has to write the word surrounded by three apostrophes (""" """). If we want to write a title, we should write the title between two equals symbols (== ==), etc. The main weaknesses of the wiki are related with the editing and formating tasks of the document:

- There is not an standard way to write documents, bold, italic, headings, etc. have different syntax in different wiki-engines so, the code cannot be saved or copied from one engine to the other without modification.
- The way how different wiki engines store information is not standard: some of them store directories for each page, some others plain files, etc.
- The *Wiki* is not appropiate, as some people could think, for storing all type of content. Debates or discussions, diaries, documents, etc. are not suitable for the wiki and using the wiki for writing such content add no value to it.
- The *Wiki* is not an auto-organized software. Users must provide that organization or, at least, some users have to review parts of the wiki and move articles in order to organize the content in an way it is easily browseable by users.

The process interacting with different wiki-engines needs an extra effort in order to translate data to an intermediate state, and from that intermediate state to the specific wiki storage.

Instant Messaging

Instant messaging allows real-time communication between two or more users, making collaboration easier and faster than traditional email messages. Teams can easily maintain on-line meetings that helps synchronization, planification and future work. Instant messaging applications are replacing old tools as *on-line chat* or *Internet Relay Chat* (IRC) because they are easier to use and provide presence**.**

Since the end of the nineties when ICQ, the first instant messaging with such characteristics, was developed, there have been many other instant messaging applications, most of them using centric or decentralized servers hosted by private companies. Some examples are: Yahoo Messenger, AOL, and the most used application MSN. In 2000, the Jabber instant messaging was released, using an open protocol. Jabber allows any user to install and maintain his own instant messaging server.

Extensible Message and Precence Protocol (XMPP) [7] is and open standard instant messaging protocol based on XML. It has been implemented as a server and as a client in almost all programming languages and Operative Systems. Google has used it as the core protocol for its instant messaging system GoogleTalk [13]

The main benefits of the Instant Messaging are:

- They state for presence. This means that users know the partner status (online, available, occupied, away...) before starting conversation, so they can choose to ask somebody else, to send an off-line message, or just starting a conversation.
- Quick interaction and collaboration between users
- Multi-chat
- All session can be logged to a file, and indexed for later review
- Most of them allow off-line messages
- Support sending files

The main benefits of XMPP based instant messaging are, in detail:

- Open implementation for many platforms
- No central point of control. Each user is able to install his own server and provide service to other users
- All servers are able to collaborate and talk to each other. A user from one server can easily talk with another user from other server.

Because of the extensible nature of the XMPP protocol, some efforts have been done in order to standardize VOIP system (Voice Over Internet Protocol). The main VOIP protocol description and implementation is *jingle* [8], that provides a set of messages in order to manage multimedia interactions between jabber clients.

Forums and Mailing Lists

Forums are used worldwide for posting questions and holding discussions. They follow the same approach as mail messages but, instead of using an specific client they allow publication using a web interface.

Although there are several applications (such as bugzilla) created specifically for announcing *bugs* (errors found in an application), the structure of the forum is sometimes used [11] in order to hold them. SourceForge, for instance, uses a forum structure where *bugs* are published by users. Developers can answer these messages providing some workaround, or giving some advice when a

bug will be fixed.

Forums are very useful tools for developers and users, so far they are giving hints about which kind of problems can arise, as well as how to solve them meanwhile they are being fixed up. If the community is large enough, and messages are answered promptly, forums become very useful. Forums are usually integrated in the most famous web-search tools (as Google, Yahoo, etc) allowing users to find and solve particular problems, even if they are only users and not part of the development team. The main advantages of the forums are:

- They can hold any kind of information, from the details of the installation procedure, to the best way to perform an action, or hints about best practices.
- Any user can ask for information or start discussions about any related topic.
- All forum messages are published along with the date when the message was released and then, it is easy to check the validity of the information.

Email is the best known tool for communications between partners, members of a team, and people in general. Almost everybody knows how to send an email, and it is probably one of the best tools for not specialized users. However, there are two main weaknesses when using it in the context of the development team:

- First of all, it is an asynchronous tool, which means that the sender of the email is never sure if the information was read or not. On the other hand, it is also one of its most successful characteristics, because there is not pressure in giving an answer, and it is not as intrusive as a phone call.
- Specifically speaking about development teams, the sender of an email has to explicitly specify the addresses of the receivers of that email. When the development team is big enough it is difficult to send an email to the whole team, because teams are growing and squeezing without user knowledge and, even when it is possible to define internal groups in the email client, the group becomes out of date.

The asynchronicity of the email can not be solved but, mailing lists, help us to cope with the difficulties of addressing to a huge and dynamic group. A mailing list is a fake email direction that acts as facade for many other emails. When an email is addressed to a list, and the list gets the email, it is resent to all the members of the team subscribed to that list.

Mailing lists have been widely used by communities, allowing the interaction among huge communities, as well as small ones. A list is set up for talking about an specific topic and users can subscribe or unsubscribe to it depending on a policy (some lists are public, other privates, etc.)

As in the case of the forums, a public mailing list usually publishes all messages in a public page that can be indexed by the most common web search engines. They have the same advantages as forums, plus the advantage that messages can be stored, read and written off-line with an email client.

News Publication

Even more than the code, documents, help pages, etc. the news are what people mostly read about a project. The blog systems provide a very easy and intuitive way to establish communication between the team and the rest of the world. Using a blog system people and members of the team can be informed about events, new releases, thoughts, etc.

Usually a set of members of the team are in charge of publish some news about the project: upcoming events, new releases, improvements, etc. that are presented as an HTML page to the rest of the world. Creators of the information can mark the information using categories (ie. Sport, Development, Java, etc) that are later used by search users in order to search for information.

We can see that users that commonly read bolgs use client programs named "news readers". These programs let them to read news from their favorite sources without the necessity of using a browser. This is because blogs are published in form of XML files that can be easily read by a client program. This ability does also allow one of the most important blogs features: the aggregation of information.

The Really Symple Sindication (RSS) is an XML format dessigned to share information between applications. This XML is easily parsed by applications, that can access directly to the information (title, category, text, etc). Most Internet sites publish information in both formats: HTML that is human readable and focused on design, colors, images, etc. and RSS format, that is this usually readed by other applications. This is an easy way for aggregation and transformation of the information, as it has been already probed within the Opaals project in two of the WP9 task “OPAALS web-office” [15, 16]

Several sites in the Internet are simply acting as catalysts, gathering news form other sources [10]. These sites usually hold a list of sites that have to be checked periodically, and a list of category words that make blogs available when they are published. During the process of checking the web sites, if information changes and any of the category words match with the internal list, the article is retrieved and stored locally. It will be part of the catalyst blog since now.

We can see main features of the blogs are:

- Easy edition of content
- People in general is informed about the development of the project, upcoming news and other information that have not to be kept as private to the members of the team.
- The blog system is though to be able to exchange and aggregate information easily.

Distribution Challenge

Here we introduce some of the approaches that can be adopted in a first instance, before the P2P framework and the transaction model is completed. This model may be incomplete and be only valid for small teams, at least at the beginning, but we hope more complex models will be achieved with the feedback from the WP3 task “P2P networking new tools and paradigms”.

Distributing the File Repository

Usually, all users involved in the development of a project download and maintain the whole repository in their hard disks. Actually, we have several copies of the same content distributed over the network. What happens if the central server collapses ? Actually nothing, because it can be reconstructed from the individual copies if, of course, all users can synchronize their efforts.

This synchronization process among users is what we want to be automatically achieved. We can set up a peer service where each user has a CVS/SVN-like server in his machine where the documents are stored. This is a simplistic view of the process where we delegate the efficiency and process time in the P2P framework. We can describe an approach to the distribution of one of the most centric-based application in three steps. The server has the following features:

1. The server can contact with all on-line users each time a new version is saved in the local repository sending an update message: "file X has been uploaded to version Y. Please, download."
2. The message is delivered to all CVS/SVN-like servers that can choose whether downloading the new version, or not.
3. Apart from the previous functionality, each CVS/SVN-like service can send a "synchronization" message to all on-line users when they log in, or the machine goes on-line, allowing for the synchronization of the content.

Could it happen that the content is lost, or gets corrupted? Of course, live usually is not as easy and, when the number of users with the content is too small, the whole content could be theoretically lost. For example, if there are only two users within the project, one living in Europe and other in America. It is very improbable that both users will be available at the same time. In this case, synchronization of contents will not happend, because when one user is connected, the oder is disconnected. Therefore, each one of them will have partial content of the whole repository, each user will have his own content as if no other user where working on it.

In such cases this can be only solved by a third user that can join the team and act as a bridge between both users. In these cases, when the amount of users is not enough (usually at the beginning of the project) it is very useful that one of the peers will be available as long as possible, let us say, during all the day. This can be achieved setting up one peer in a 24 on-line computer. This peer will act as a bootstrapping peer.

Setting up a CVS/SVN-like service always on-line and available is an easy way to maintain a fresh copy of the repository, with the advantages of central services, as well as the advantages provided by the P2P architecture:

- Setting up a second CVS/SVN-like service is an easy way to have a backup of all the data without the need of special programs or difficult configuration.
- Provide a farm of repositories is as easy of setting up different CVS/SVN-like services in different machines.

- And all the typical P2P advantages, so far no configuration is needed, no fix IP needed, no expensive frameworks, minimum bandwidth needed, etc.

Let us take this measure as temporary, and only necessary at the beginning of the creation of a new community. When the community is large enough, peer will connect and disconnect but often, if not always, there will be several peers on-line at the same time, making this replication process possible without the intervention of any central repository. However, the P2P architecture must never rely on the assumption that a node will be available twenty four hours a day.

Distributing the Wiki

The most famous *wiki* site is the *Wikipedia*, namely, the most visited and comprehensive *free encyclopedia* [6]. The English version, with almost two million definitions, is about one Gigabyte of compressed information, supporting one hundred thousand update operations a day. But if we take into account the history of each work, the storage needs are really over the top. Only the definition of "Jesus", for instance, has a history information saved of about eight hundred Megabites, "United States" about one Gigabyte or "George W. Bush" more than one and a half Gibagyte.

In order to maintain the hardware and the software infrastructure able to support such amount of data and users, a huge amount of money, machines and manpower it is necessary. Most of the projects, of course, do not need such a quantity of resources, but when we want to be the catalyst of other research projects' (like Opaals want), the distribution of the *wiki* using a P2P architecture have to be considered. Principal advantages of the distribution are [4]:

- improve performance
- improve reliability
- reduce bandwidth and storage cost
- allow aggregation.

The last advantage, aggregation, is specially relevant for a project like Opaals, to become the meeting point for other research projects. Projects do not have to be forced to rely on an external provider, but rather they can still maintain their own P2P architecture (corporate machines or individuals), providing content for the Opaals wiki, enjoying at the same time the benefits from other *wikis*.

We start form the fact that we have successfully developed a way to maintain (create, retrieve, update and delete) a file repository. In that case, we can regard the *Wiki* as a file-interpreter. The Wiki is using files for storing and showing information. Therefore, we propose to create a distributed Wiki over a distributed file repository structure, as already achived with *Code co-op*, comercial software [14].

Using the distributed file reposytory explainded in the previous section, we can manage to share wiki-pages among users.

1. The server can contact all on-line users each time a new page is created/modified/deleted sending the following message: "page X has been modified. New version is Y"
2. When users get the message, they can update their index of pages and, if necessary, download the new version of the page.
3. Apart from previous functionality, each Wiki-like service can, send a "synchronization" message to all on-line users when it starts, or the machine goes on-line, allowing synchronization of the indexes.

Note that this is the same simplistic approach as before, but with some interesting modifications. In the purest P2P fashion, each node can have an index accounting for a few available pages in the global wiki, so it can maintain an updated cache. This is fundamentally a first improvement for saving

bandwidth and time. Having information of all available pages in the global wiki also allows to perform local search.

Recently, we have found two very new research projects focused on finding the solution to the same Wiki decentralization process [3, 4]. It will be, for sure, a good collaboration opportunity.

Distribution Challenge IM

The nature of Jabber instant messaging and the XMPP protocol allow an easy integration of different domains. As stated above, different servers can interact one each other, allowing inter-domain user interaction.

Two different instant messaging servers can be easily integrated, as far as both servers use the XMPP protocol as Instant Messaging application.

The integration performed by Jabber servers is simple and it relies on the fact that user names (in Jabber) are a composition of the name of the user and the domain of the server hosting the server. Messages are send to the user server with the command TO: user@domain. The server redirects the message to the domain that will deliver it to the user.

Distribution Challenge Forums and MailingLists

In both cases the distribution is possible, mainly using the three techniques described above. Forum messages can be stored as simple files (emails).

In this case we want to use some distributed cache system, where nodes store static content during a certain time. Note that this distributed cache system makes sense when we talk about forums messages or emails that are, by nature, non editable. This means the content of an email can be replied, forwarded or erased by an user, but never changed. The content is able to be cached by nodes, following a probabilistic rule. Content most retrieved will be available more time.

This approach means that the most queried and demanded content will be spread through the network (or, at least, through the surrounding the nodes of the information provider) for a certain amount of time proportional to the importance of the information, hoping that new information will be available in the network for more time than obsolete information, thus allowing users to synchronize quickly.

Another advantage of these kind of application regarding the exchange of information is that emails and forum messages do not need to be fully downloaded, but only headers. In addition, people is usually subscribed to a finite number of forums, because makes no sense to be aware of everything happening in the universe. In most of the cases only the header synchronization will be enough. Only people involved in discussion will download, read and respond messages.

Installed Applications

We chose applications based on three different premises:

1. The project has to be an Open Source project with GNU-like license, allowing to modify the code
2. The programming language has to be easily used in as many platforms as possible: Linux, Windows, Mac and embedded devices
3. If possible, we avoid the use of databases, relaying in flat files as the better way to achieve distribution of content.

Following these rules, we chose these tools:

Description	Application	Language	Flat files	URLhttps://www.opaals.org/svnroot/
File Repository	SVN	C	yes	https://www.opaals.org/svnroot
Wiki	MoinMoin	Python	yes	http://wiki.opaals.org
Blog	Frog	Python	yes	http://blogs.opaals.org
IM	WildFilre	Java	yes	http://opaals.org:5222
Mailing List	Mailman	Python	yes	http://lists.opaals.org
Forums	phpBB	Python	no	http://forums.opaals.org
WebDav	mod_webdav	C	yes	http://files.opaals.org

The integration of all these applications caused few changes and new developments in order to integrate them. We made a new registration system where users submit their request and the Opaals coordinator accept them. Once accepted, a new password was auto-generated and then, the registration system, registers the same user, with the same pasword in all listed applications. The username and a hash of the password is stored for further development. We could, for instance, add the webdav system (WP9 and WP10) without the need of registering users again or change their password.

All tools are accessible using the Opaals website (http://www.opaals.org), except the MailingList, that the consortium decided to keep private. Only the project coordinator has the rights to add people to the mailing list, at least in this first phase.

The list of changes done can be divided in two different groups: integration of applications, integration with external applications, improvements

Integration of applications

The integration of the application within the same server lead to some changes:

- scripts for registration in each tool
- scripts for deregistration in each tool
- set of scripts for creating a new user, auto-generate passwords and welcome email
- integration with the email system
- installation and set up of all requiered packages for each tools, wich includes: Apache, Python, MySQL, PHP, Graphic processing tools, Java and different servers
- set up security and special security policies as IP restriction in the access to applications and remote backup system

Integration with external applications

Along with the installation of these tools, the OKSDesktop [16] was already developed, as a part of the WP10. In order to allow the interanction between the OKSDesktop and the set of tools, an authoring mechanism and a set of services where developed.

We developed some bots that act as remote servers. Bots are non-human chat users that understand some questions and reply to them. In our development, bots understand some actions and then execute command in the remote computer (create new user, get file, etc). Bots use Jabber communication and then only talk with authorized users using an encrypted stream.

Some of the changes in the platform were:

- authentication using Jabber
- set up several bots for file exchange, creation of users, remote communication, etc.

Improvements

Specific development was needed in order to fulfill the request of part of the consortium. Most of the new development to provide new functionallity was related with the wiki and the visualization of the information: chars, graphs, tagging system, etc. Some examples of the changes can be found at these URLs:

- Visual site map: http://wiki.opaals.org/OPAALS_Home?action=VisualSiteMap
- Ontology: http://wiki.opaals.org/?action=ontology

Conclusions and future work

During the first months of the project we have provided an Open Source platform including the most important tools used nowadays for software development and user interaction. This first set up of the tools allow people within the project to create content, foreseeing to some extent how collaboration emerges, how can it be guided, and even more.

Our main goal for the next months is to be able to break the dependency with centralized applications.

We are willing to face up the exciting job of distribution and aggregation that relies on the P2P architecture. Other workpackages (ie. WP3) are already providing theoretical knowledge about how to design the distribution. We will work together with other workpackages (ie. WP9 and WP10) in order to achieve distributed goals during the next months.

The process of distribution will start with the file repository that will become the basis for other applications, followed by the wiki and the blog system. For such a task we have asked other partners involved in the project (LSE, UNIS, TechIDEAS) to provide and set up a computer with Internet availability, accesible from outside of the institution. These computers will become the core of the P2P network and will hold most of the information that is now stored in the central server http://www.opaals.org

References

1. Klingenstein, Ken. (2003). The rise of collaborative tools. Educause Review, 38(4): 60-61.
2. J.E. Robbins, "Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools," in Feller, J., Fitzgerald, B., Hissam, S.A. and Lakhani, K.R. (Eds.) *Perspectives on Open Source and Free Software,* The MIT Press, Cambridge, MA, 2004.
3. Guido Urdaneta, Guillaume Pierre and Maarten van Steen. "A Decentralized Wiki Engine for Collaborative Wikipedia Hosting". In *Proceedings of the 3rd International Conference on Web Information Systems and Technology* (WEBIST), March 2007.
4. P2P Wiki http://www.cs.bsu.edu/homepages/chl/P2PWiki/
5. Ohira, M., Ohsugi, N., Ohoka, T., and Matsumoto, K. 2005. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings of the 2005 international Workshop on Mining Software Repositories* (St. Louis, Missouri, May 17 - 17, 2005). MSR '05. ACM Press, New York, NY, 1-5. DOI
6. http://wikipedia.org
7. RFC 3920 "Extensible Messaging and Presence Protocol (XMPP): Core". http://www.ietf.org/rfc/rfc3920.txt
8. XEP-0166: Jingle (http://www.xmpp.org/extensions/xep-0166.html)
9. http://jabber.com
10. http://planet.maemo.org
11. http://www.sourceforge.net
12. http://java.net
13. http://www.google.com/talk/
14. Code Co-op, distributed version control system. http://www.relisoft.com/co_op/help/index.htm
15. T. Desodt. "Stable OPAALS Web Office". Deliverable D9.3. Opaals
16. T. Desdodt. "1st OPAALS Newsletter", Deliverable D9.5.1. Opaals