PaSaMuF Documentation

PaSaMuF Documentation

Table of Contents

About this document ....
I. The Requirement Specification for PaSaMuF Peer-to-peer Filesharing tool. ....
1. Introduction ....
2. First thoughts concerning requirements ....
3. The Purpose of the Product ....
4. Client, Customer and other Stakeholders ....
5. Users of the Product ....
6. Mandated Constrains ....
7. Naming Conventions and Definitions ....
8. Relevant Facts and Assumptions ....
9. The Scope of the Work ....
10. The Scope of the Product ....
11. Functional and Data Requirements ....
12. Look and Feel Requirements ....
13. Usability Requirements ....
14. Performance Requirements ....
15. Operational Requirements ....
16. Maintainability and Portability Requirements ....
17. Security Requirements ....
18. Cultural and Political Requirements ....
19. Legal Requirements ....
20. Off-the-Shelf Solutions ....
21. New Problems ....
22. Tasks ....
23. Cutover ....
24. Risks ....
25. Costs ....
26. User Documentation and Training ....
II. Developer Documentation ....
27. Architecture ....
28. The Classes ....
Indexer Classes 37
class Indexer 37
class MetaDataInfoObject 38
class FulltextExtractor 40
Core Classes 41
class CoreService 41
class MasterServerConnection 42
GUI Classes 45
class StartServlet 45
class PreferencesServlet 45
class ErrorServlet 45
class GuiTemplate 46
MasterServer Classes 46
class MasterServer 46
class PeerAddress 47
class PeerAddressList 47
Utility Classes 48
class pasamufProperties 48
class pasamufErrorLogger 49
class FileScout 49
29. Diary ....
30. The Sub-Projects and Team Members ....
Project Management 53

Marc Aßmann 53
User Interface (Java, Servlets, Webservices) 54
Anja Nikoleit 54
Roland Brackmann 55
Application Core (Java, Servlets, Webservices) 55
Richard Metzler 55
Lars Trieloff 55
Martin Probst 55
Benedikt Meuthrath 56
Master Server (Java, Servlets, Webservices) 56
Alex Saar 57
Jan Hartmann 57
Johannes Wust 57
Indexing Module 57
Alex Klimetschek 58
Florian Brodersen 59
Ole Weidner 59
Documentation 59
Thomas Wendlandt 59
Testing 60
Conclusion 60
III. User Documentation .....
31. Installation .....
Installation Prequisites 62
Compiling and Installing PaSaMuF 62
Step by Step Installation of PaSaMuF client 62
Step by Step Installation of PaSaMuF MasterServer 63
32. Running PaSaMuF .....
The Structure 65
Changing the Preferences 65
Searching 66
Simple Search 66
Extended Search 67
My Documents 68
Downloading Files 68
Lucene Query Syntax 68
Fields 69
Term Modifiers 69
Wildcard Searches 69
Fuzzy Searches 70
Proximity Searches 70
Boosting a Term 70
Boolean operators 70
OR 71
AND 71
+ 71
NOT 71
- 71
Grouping 72
Escaping Special Characters 72
A. Contributing to this document .....
Docbook editing guidelines 73
B. Glossary .....
C. License .....
The Apache Software License, Version 1.1 77
GNU General Public License 78
Preamble 78
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 78

List of Figures

9.1. context diagram 14
10.1. Product Boundary 16
10.2. The application core should work as shown in the petri net. 16
27.1. Architecture Draft 35
28.1. class Indexer 37
28.2. class MetaDataInfoObject 38
28.3. class FulltextExtractor 40
28.4. class CoreService 42
28.5. class CoreService 43
28.6. class StartServlet 45
28.7. class PreferencesServlet 45
28.8. class ErrorServlet 45
28.9. class GuiTemplate 46
28.10. class MasterServer 46
28.11. class PeerAddress 47
28.12. class PeerAddressList 47
28.13. class pasamufProperties 48
28.14. class pasamufErrorLogger 49
28.15. class FileScout 50
32.1. An example for preferences 65
32.2. Simple Search 66
32.3. Extended Search 67
32.4. An example of search results 68

About this document

The documentation will include the definition of requirements, the evolution of the project, what the team members learnt during the project, what they would like to do different in further projects, an end user documentation and, last but not least, a glossary discribing some terms that may not be clear to each and every person reading the the documentation.

For contributing text to this document see the appendix “ Contributing to this document ”.

Part I. The Requirement Specification for PaSaMuF Peer-to-peer Filesharing tool.

The requirements specified within this document, except the very first thoughts, are structured by requirement types orienting on "Volere - Requirement Specification Template (Edition 6.1)" by James and Suzanne Robertson:

Functional requirements are the fundamental subject matter of the system and are measured by concrete means like data values, decision making logic and algorithms.

Non-functional requirements are the behavioral properties that the specified functions must have, such as performance, usability, etc. Non-functional requirements can be assigned a specific measurement.

Project constraints identify how the eventual product must fit into the world. For example the product might have to interface with or use some existing hardware, software or business practice, or it might have to fit within a defined budget or be ready by a defined date.

Project drivers are the business-related forces. For example the purpose of the product is a project driver, as are all of the stakeholders - each for different reasons.

Project issues define the conditions under which the project will be done. We include these in the requirements specification to present coherent picture of all the factors that contribute to the success or failure of the project.

All requirements that are to be validated within the testing and validation process have a priority assigned: [P1], [P2], [P3]. [P1] - Prioritylevel 1 - These requirements must be successfully validated in order for the product to function at all. [P2] - Prioritylevel 2 - These requirements represent essential functions. However unsuccessful validation should not cause the product to fail in total. [P3] - Prioritylevel 3 - These requirements represent additional functions. Eventual dependencies and conflicts are specified within the respective requirement specification.

Chapter 1. Introduction

The software will be a file sharing tool for small communities up to thirty persons. It should be specialized on office document formats to support collaborative work.

One feature: remote control of a client is still being discussed.

Other requirements that are still in discussion, can be found in the project's forum [http://www.marc-assmann.de/kdp/phorum/list.php?f=11].

Each requirement in this chapter will be a section. The original submitter of the requirement will be marked as author. This has to appear in the section info. Please do not forget to assign an id to each and every requirement, to make cross-referencing possible.

Chapter 2. First thoughts concerning requirements

- **Share files.** The user can specify a folder to share with others. All documents of this folder can be accessed by other peers.
- **Search for files.** The search interface should allow to search on all peers for documents containing a key-word. An advanced search interface should allow users to specify things like filetype, date,...
- **Index the files to be shared.** The shared files should be indexed by
  - date
  - size
  - name
  - pathinfo
  - fulltext
  - author

  and more.
- **"Master-Server", which keeps track of who is online .** The master server should allow one client to log on/off the network. This will enable it to distribute a list of currently connected clients.

  ### Network health

  For network health, it must make sure the list is up to date.
- **Reliable Network.**
  - Clients can connect to the network any time.
  - Clients can disconnect from the network if they are connected.
- **Accessible User-Interface.** The User-Interface should be accessible and lightweight. A web-browser based solution is preferred.

Chapter 3. The Purpose of the Product

a. **The user problem or background of the project effort.** Knowledge workers produce and consume large amounts of documents. They are in need of effective ways to access documents written by other knowledge workers.

b. **Goals of the product.** The product will speed up the document-flow in small knowledge-worker communities. Every member will have access to all documents shared by community members. The system should evolve as a Knowledge Management platform in the future. Therefore, it must heavily rely on open standards. This will enable feature-extension to the system.

Chapter 4. Client, Customer and other Stakeholders

a. **The client.** is the person/s paying for the development, and owner of the delivered system. As far as our project is concerned, the clients are equal to the project members [P1]. There is no one paying for the development.

b. **The customer.** is the person/s who will buy the product from the client. Until further developments, apart from the project members [P1], only HPI students are going to use PaSaMuF, but they will not have to pay for it.

c. **Other stakeholders.**

Kader Sarahoui, evaluating professor [P2]

- Knowledge needed by the project: Requirements Engineering, general support.
- Necessary degree of involvement: general involvement, evaluation of the system.

The open source community [P1].

- **Knowledge needed by the project:** The open source community provides many solutions to problems that have to be solved by the system.
- **Necessary degree of involvement:** No involvement in development.

The users

- (see “Users of the Product”)

Chapter 5. Users of the Product

a. **The users of the product.**

The project members [P1].

- User name: "key users"
- User role: client, customer, developer, primary target user
- Subject matter experience: as developers of the system every group member masters the subject matter.
- Technological experience: all group members are well experienced.
- Other user characteristics:

HPI students [P2].

- User name: "secondary users"
- User role: secondary target users. Besides project members, this group will make most use of the system.
- Subject matter experience: good knowledge of the subject matter can be assumed.
- Technological experience: above-average knowledge of the technology can also be assumed.
- Other user characteristics:

Other users [P3].

- User name: “unimportant users”
- User role: do not belong to any target group.
- Subject matter experience: most probably good knowledge.
- Technological experience: most probably sufficient knowledge to master the system quickly.

b. **The priorities assigned to users.**

- Key users: The project members have maximum priority.
- Secondary users: All issues that are not decided because of key user priority will be set to fit secondary user requirements.
- Unimportant users: All other users may not expect any support.

c. **User participation.**

- Key users: As project members, these users will directly participate in the development process.

- Secondary users: Some members of this group will be asked for feedback on the system design, therefore gaining indirect influence on the system development.
- Other users: As the project strongly focusses on its target users all other users do not participate in any way.

Chapter 6. Mandated Constrains

a. **Solution constraints.** The system should run on all platforms commonly used by knowledge workers: [P1]

- Windows (2000/XP)
- Linux
- MacOS

It will therefore be implemented in Java. The system should support heavily used file formats. Especially the following: [P1]

- Pdf Documents
- Textfiles
- MS Word
- MS Excel
- MS Powerpoint

All supported file formats (except for Textfiles) must be searchable by meta-information: [P1]

- author
- title
- creation date

The system must also provide full text search for textfiles and Pdf-Documents. [P1] The system should have an easy to use graphical interface. [P1] Userinterface language: English [P1]

b. **Implementation Environment.** PaSaMuF has been built in both Linux and Windows operating systems and will be running on both systems, thanks to Java's portability.

c. **Partner applications.** The system will use an open source Java Servlet Engine. The goal is to find an easily embeddable and stable solution. Etymon™ PJ (http://www.etymon.com/pj) will be used for PDF-Extraction.

The system must be compatible to common webbrowsers:

- MS Internet Explorer v4.0 and above [P1]
- Netscape Navigator v4.0 and above [P1]
- Opera v6.0 and above [P1]

d. **Commercial off the shelf packages.** The system will completely rely on freely available tools [P1].

e. **Anticipated workplace environment.** The system is designed to work wherever the hardware it is running on works. For a specification of minimal hardware requirements see “Expected technological environment.”. Especially following workspace constraints must be considered: - The workspace might not support audio signals! [P2]

f. **How long do the developers have to build the system?**

- **Deadline for project components:**
  - Masterserver: 06-06-2002 | deadline fullfilled
  - Indexing Module: 13-06-2002 | postponed to 24-06-2002
  - Client Core: 19-06-2002 | postponed to 26-06-2002
  - GUI: 24-06-2002 | postponed to 30-06-2002
- Deadline for test begin of system as a whole: 01-07-2002
- Deadline for documentation: 01-07-2002 to 10-07-2002
- Deadline for the project: 01-07-2002 to 11-07-2002

g. **What is the financial budget for the system?** As this is a project for studying purposes, we do not have any financial budget.

Chapter 7. Naming Conventions and Definitions

- **project members:** the following people:
  - Marc Aßmann
  - Florian Brodersen
  - Roland Brackmann
  - Jan Hartmann
  - Alex Klimetschek
  - Richard Metzler
  - Benedikt Meuthrath
  - Anja Nikoleit
  - Martin Probst
  - Alex Saar
  - Lars Trieloff
  - Ole Weidner
  - Thomas Wendlandt
  - Johannes Wust
- **system installer:** the utility to deploy the software onto any system that fits the "Expected technological environment"-requirements.
- **shared directory:** the directory containing the files to be shared to which other peers of a data exchange network are granted access.
- **shared folder:** to be stored by the system user.
- **the client:** 1. the system component, which enables the user to interact with peers in the network. 2. the person or institution who ordered the product and according to whose ideas and requirements it is developed.
- **peer:** a currently logged in client other than oneself.
- **Master Server:** the system component managing the system network, i.e. the server holding the list of peers actually connected to a network, and some further specified information concering each peer..
- **system network:** the set of all clients, which can possibly login, plus the Master Server.
- **target group:** group of people who, according to market analysis, should be most interested in the product. Hence, the producer strictly orientates at the formerly defined target group for any improvement or changes of the product.
- **project members:** the group of 14 HPI students directly involved in the development of PaSaMuF.

- **HPI students.** are the people studying Software Engineering at Hasso Plattner Institute. During the development of PaSaMuF, they include 3 generations (Jahrgänge), which means about 300 students.
- **Target users.** are most probably interested in using the product and whose preferences are therefore of most interest for the development of a product.
- **Networking.** is the use of data connections for all kinds of informational transfers. Information nets vary from Local Area Networks up to the World Wide Web. In this special context, only the transfers between PaSaMuF components are meant.

Chapter 8. Relevant Facts and Assumptions

a. **External factors that have an effect on the product, but are not mandated requirements constraints.** .

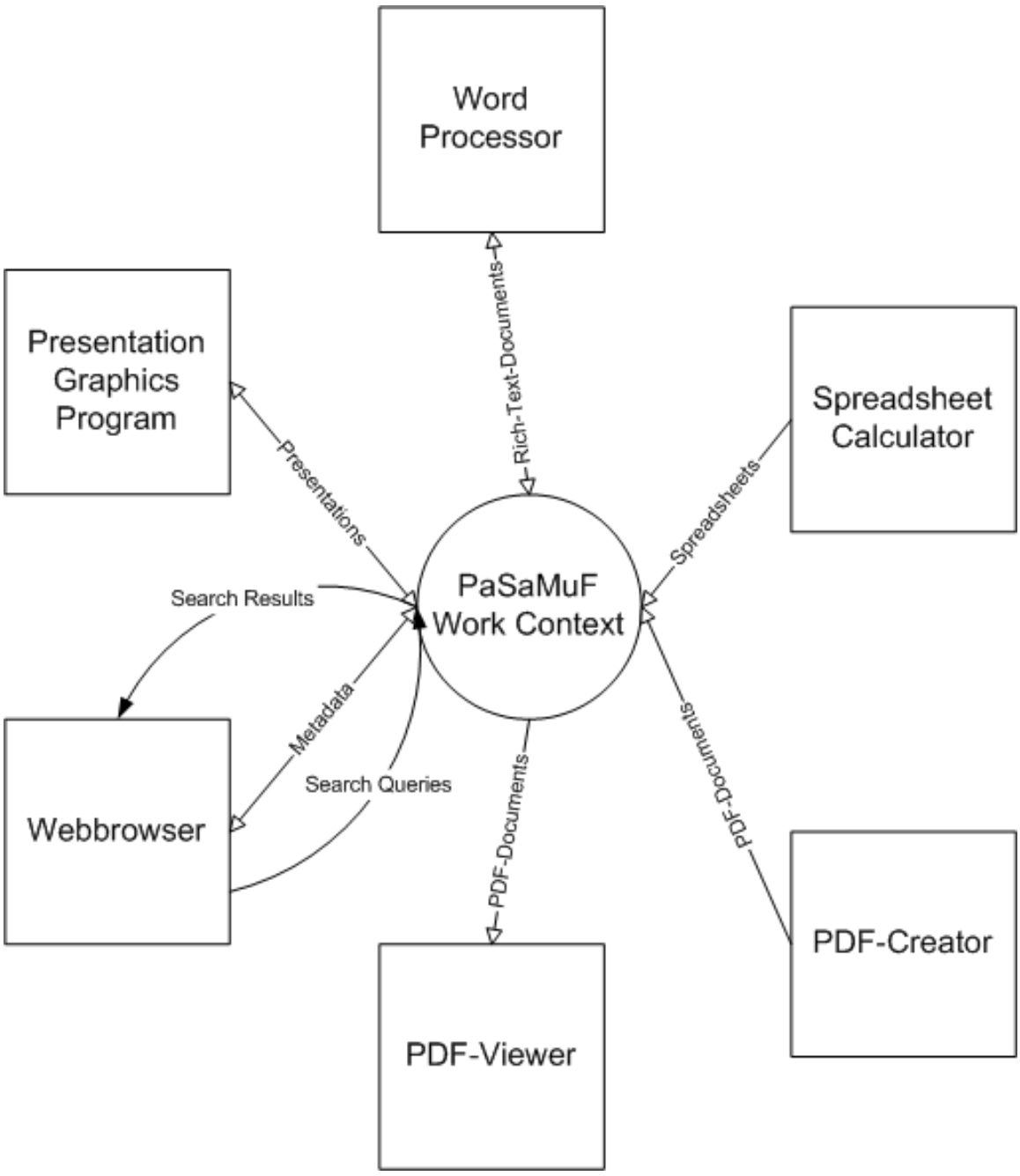
- An organization might show interest in a further development of PaSaMuF for special purposes and offer a financial budget.
- computing performance might increase in an unimaginated extend which will result in better work experience with the product.

b. **Assumptions that the team are making about the project .** All project members consider PaSaMuF, as planned, to be useful. However, a market research giving an idea of the potential success has not been performed. Therefore no assumptions exceeding the personal and HPI-wide use of PaSaMuF have been formulated by now.

Chapter 9. The Scope of the Work

a. **The context of the work.**

Figure 9.1. context diagram



b. **Work Partitioning.** Most important in PaSaMuF's work context is the web browser, which transmits the search queries. Various document creation and editing programs might also belong to the work context, either providing documents or being able to work with them. Among these might Writing applications like

MS Word, Presentation Applications like MS Powerpoint, the Adobe Acrobat Reader, but OpenOffice Applications as well.

Chapter 10. The Scope of the Product

a. **Product Boundary.** A PaSaMuF client is able connect to the network by accessing the list of clients, The user modifies the list of files by instructing the client to share files and editing file meta information to be searched. Last but not least, the user can instruct the client to search for files.

Figure 10.1. Product Boundary

b. **Use case list.**

Figure 10.2. The application core should work as shown in the petri net.

Chapter 11. Functional and Data Requirements

a. **Functional Requirements.**

1. **The Client.**

1. **Networking.**

a. "Login to the network" [P1]

b. "Logout of the network" [P1]

c. "Obtain a list of peers online" [P1]

d. "Establish connections to peers in the network as client" [P1]

e. "Allow connections to peers in the network as host (server)" [P1]

2. **Sharing.**

a. "Grant access to files in a shared directory for download by peers" [P1]

b. "Retrieve shared files of peers" [P1]

3. **Searching.**

a. "Index files to be shared for efficient searching" [P1]

b. "Send search requests to peers and retrieve results" [P1]

c. "Receive search requests and send corresponding results" [P1]

4. **GUI.**

a. "Userinterface to set Masterserver location" [P2]

b. "Userinterface to set shared folder path" [P2]

c. "Userinterface for sending search-requests" [P1]

d. "Userinterface to display search-results" [P1]

e. "Userinterface to retrieve requested files" [P1]

2. **The Master Server.**

a. "Keep track which clients are online" [P1]

b. "Send a list of clients online on requests of valid users" [P1]

c. "Add users to the network" [P1]

d. "Remove users from the network" [P1]

Chapter 12. Look and Feel Requirements

a. **The Interface.** The interface should be

- selfdescribing [P2]
- easy to use [P1]
- fast. [P2]

There is no need for graphical gimmicks.

b. **The style of the product.** The product should appeal to HPI students.

Chapter 13. Usability Requirements

a. **Ease of use.** The product should be easy to use not only for HPI students but also for people who are just used to other p2p-tools. [P1]

b. **Ease of learning.** HPI students should be able to use the product within 5 minutes productively, people who are not that well used to standard software within 10 minutes at most. [P1]

Chapter 14. Performance Requirements

a. **Speed requirements.** The network should deliver search results within a minute. Most client-configuration changes should take effect within 10 seconds. [P2] Changes in network preferences might require a restart of the application. [P3]

b. **Safety critical requirements.** Only registered users should be able to connect to the system network and to access shared files. [P2]

c. **Precision requirements.** Search results that are received from peers may not be up to date. They should be no older than 1 hour. [P2]

d. **Reliability and Availability requirements.** The system should make no assumptions about other peers availability except for the master server. [P1] The master server must be reliable as clients cannot connect to the network without it. Assuming working hardware it should achieve 98% uptime. [P1]

e. **Capacity requirements.** The product is optimized for small communities. Therefore, the requirements listed here cannot be exceeded by large amounts.

The system should

- be able to handle 30 peers online at a time [P2]
- enable every user to do send a search-request at any time [P2]
- enable a user to share up to 1000 documents [P2]

Chapter 15. Operational Requirements

a. **Expected physical environment.** Anywhere. The workspace environment requirements have to be fullfilled. [P1]

b. **Expected technological environment.** The computer running the system should fulfill following requirements:

- Processor: >= 350Mhz
- RAM: >= 30MB free
- Harddisk: >= 10MB free (+ space for files to be shared)
- Network Connection: >= 64 kbps

c. **Partner applications.** The product must interface with all operating systems supported [P1] and with proxy-server systems [P3].

Chapter 16. Maintainability and Portability Requirements

a. **How easy must it be to maintain this product? .** Add user to network within 15 minutes. [P2] Remove user from network within 15 minutes. [P2]

b. **Special conditions that apply to the maintenance of the product.** PaSaMuF is a unique project work belonging to the lecture “Concepts and Algorithms” by Kader Sahraoui. Further developments are not yet intended. However, if PaSaMuF has proved to be useful, HPI students are free to improve it according to the General Public License. Very specific extensions needed for business purposes might also be commercially distributed...In this case, capital investments might become necessary.

c. **Portability requirements.** For a list of operating systems the system is to run under, see “Solution constraints”.

Chapter 17. Security Requirements

a. **Is the system confidential?** The system is confidential to a certain degree. Only registered members are to be able to connect to the system-network. [P1] However, search requests as well as file transfers will not be encrypted, therefore not eliminating the possibility of being intercepted by non-registered people. If time permits, encryption will be implemented. [P3] The system is not to be utilized for highly confidential material!

b. **File integrity requirements.** Invalid Files will not be found.

Chapter 18. Cultural and Political Requirements

Must confirm with “The HPI Mission” [P1].

Chapter 19. Legal Requirements

a. **Does the system fall under the jurisdiction of any law?** As the system is meant for sharing of documents, users are responsible not to share documents they don't own the copyright for. Users downloading documents from other peers have to respect the authors copyright according to international scientific standards. That means, they are allowed to read and cite those documents.

b. **Are there any standards which we must comply?** none.

Chapter 20. Off-the-Shelf Solutions

a. **Is there a ready-made system that could be bought?** Existing file-sharing applications could be used for file-sharing purposes, but all of these neither provide searching meta-data nor full-text search. These applications include:

- KaZaa
- Limewire
- Bearshare
- WinMx

b. **Can ready-made components be used for this product?** Following ready-made components can be used for this product:

- Apache Tomcat Servlet Engine [http://jakarta.apache.org/tomcat/index.html] as underlying application environment for networking.
- Webservice Toolkit [http://xml.apache.org/axis/index.html] to integrate webservices
- Apache POI [http://jakarta.apache.org/poi/index.html] to extract information out of documents
- Apache Ant [http://jakarta.apache.org/ant/index.html] as deployment/install tool
- Apache Lucene [http://jakarta.apache.org/lucene] to index metadata and text for fast searching
- Etymon PJ [http://www.etymon.com/pj] to extract data from PDF files

c. **Is there something we could copy?** Following systems provide similar functions regarding file-sharing techniques:

- Napster
- WinMx
- Audiogalaxy
- KaZaa
- Limewire
- Bearshare

These systems also include functional GUIs that are of interest.

Chapter 21. New Problems

a. **What problems could the new system cause in the current environment?** The system will have a security footprint as it needs an open HTTP port on every peer. The security will therefore rely on the used 3rd party applications.

b. **Will the new development affect any of the installed systems?** The system is to stand alone and should not affect any other installed systems. However, firewall systems may have to be reconfigured to allow the system to function properly.

c. **Will any of our existing users be adersely affected by the new development?** There are no existing users.

d. **What limitations exist in the anticipated implementation environment that may inhibit the new system?** If technological requirements are not fulfilled, performance problems might be caused.

e. **Will the new system create other problems?** none known.

Chapter 22. Tasks

a. **What steps have to be taken to deliver the system?** The system will be available for download for the primary target usergroup. [P2] Secondary target users will be notified by email that they can acquire the system from primary target users. [P2]

b. **Development phases.** Assuming a functional technological workspace, the system can be integrated in the operating environment by executing the system-installer.

Chapter 23. Cutover

a. **What special requirements do we have to get the existing data, procedures to work for the new system? .** There are no existing system components. Existing data (knowledge-files) that are to be shared must be copied into the shared directory.

b. **What data have to be modified / translated for the new system?** none

Chapter 24. Risks

- The system as a whole may be too slow and disturb the users work. This would surely lower the acceptance of the system significantly.
- Important developers may stop being available for the project, which would be difficult to compensate within project time. This holds especially true for members planning and implementing the application core and the team realizing the indexing module.

Chapter 25. Costs

For all of the team members, the project costs a lot of time. However, they are willing to sacrifice this time as they know they will profit from it.

Chapter 26. User Documentation and Training

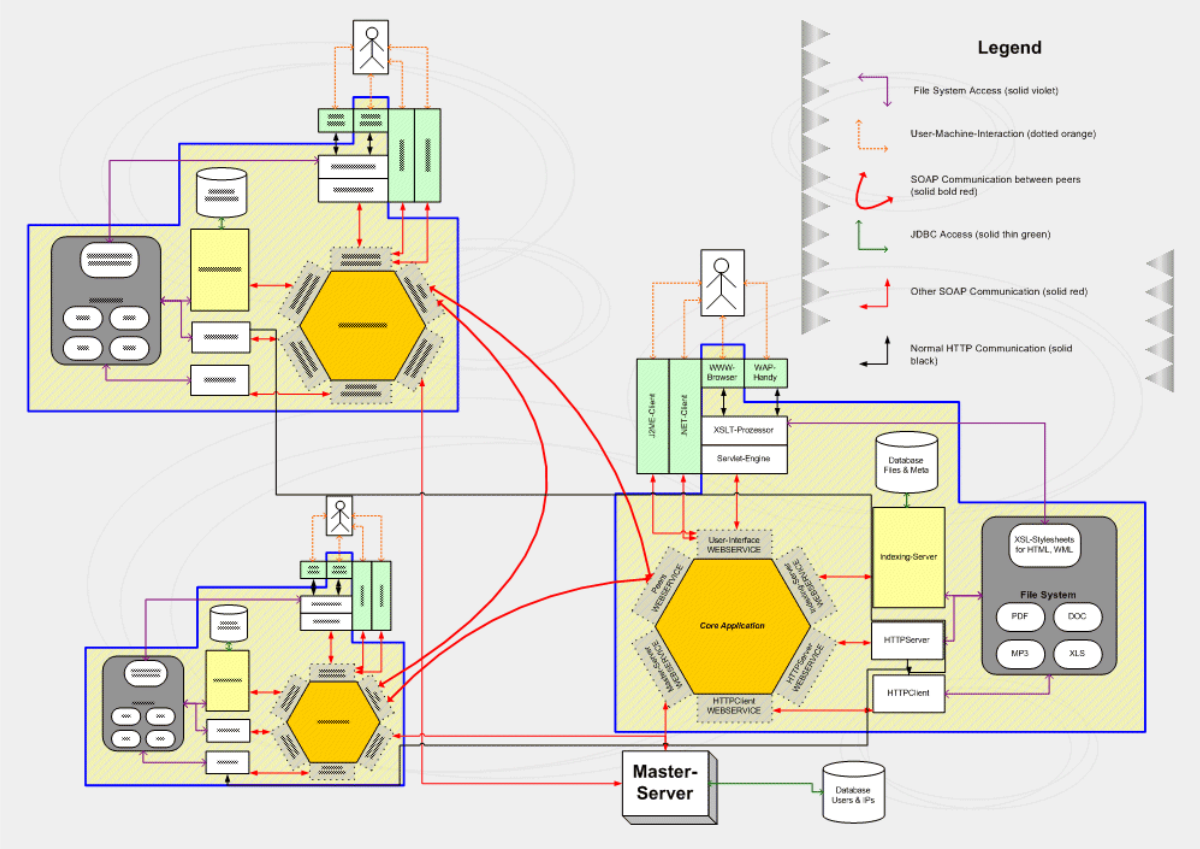
First of all, the user documentation should not be necessary at all for HPI students. Less experienced users should need the documentation only very rarely, when very specific information is needed, or for parts of usage that exceed the most common software functions.

Part II. Developer Documentation

The second part of this book is reserved for the developers' documentation. It includes architecture drafts and class descriptions.

Chapter 27. Architecture

Figure 27.1. Architecture Draft



The central part of the application is called core application. It manages all communication with peers as well as coordinating user input and steering external components.

The core application of each client has a native java interface to an indexing service, which accesses the shared folder and extracts meta-data from documents. Afterwards this metadata is stored in a database.

The user can control the client over a HTML-based GUI which is generated by a servlet which communicates with the client over a SOAP-Webservices interface. This interface is also open for other GUIs like J2ME or MS .NET clients.

Each client registers at the master-server. The master server knows who is online and has the ip-addresses of every client in memory. If a new client connects to the network, the master client can tell him the IPs of his peers. Therefore a SOAP-interface exists which the client can use to communicate with the master-server.

The peers themselves also communicate over webservices. So they can search for files and tell each other the results.

If an appropriate file has been found, the HTTP-Client of the core application connects to its peer's HTTP-

Server and receives the requested file.

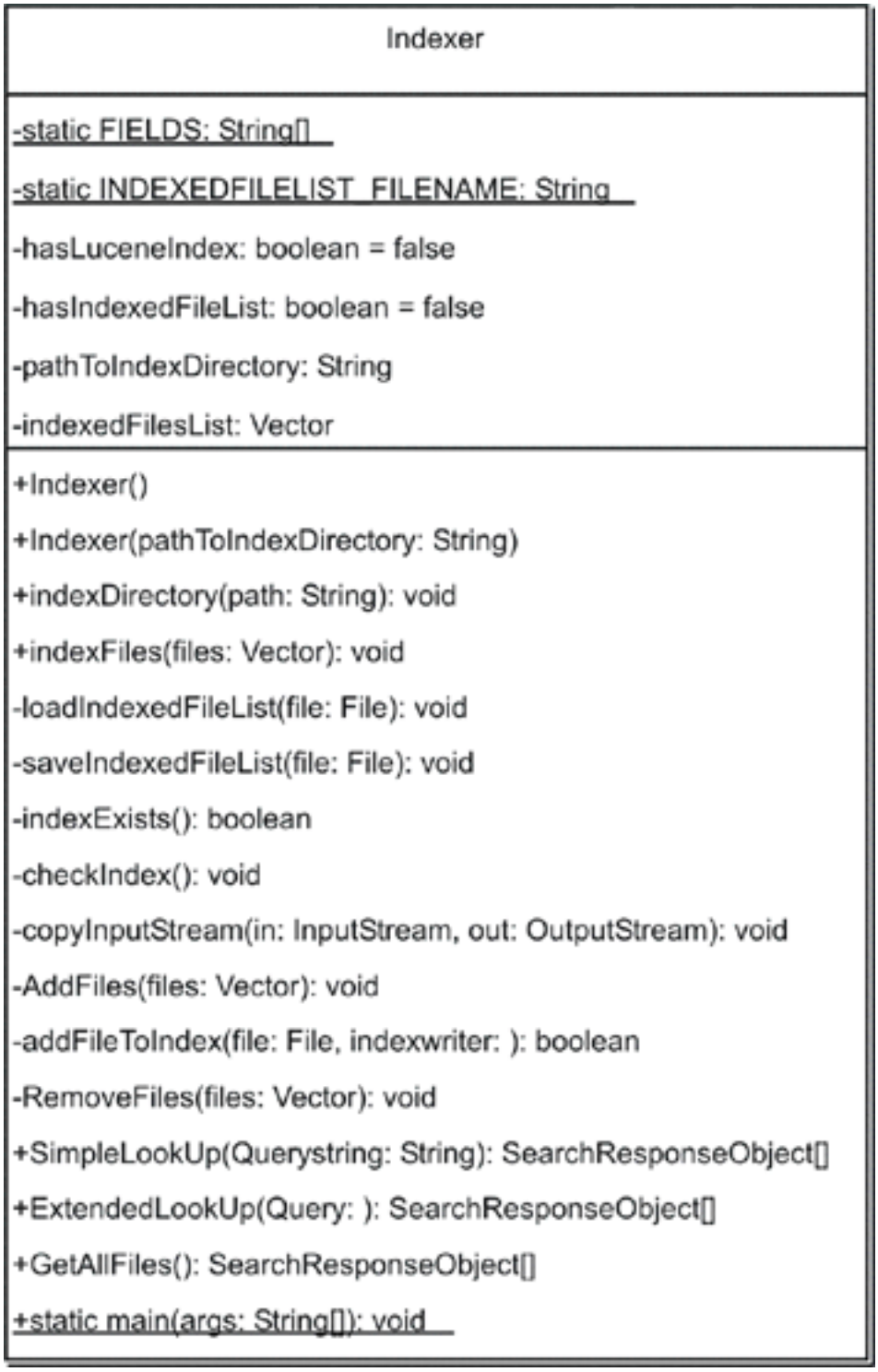
Chapter 28. The Classes

Indexer Classes

class Indexer

This class is the core and interface of the indexing module.

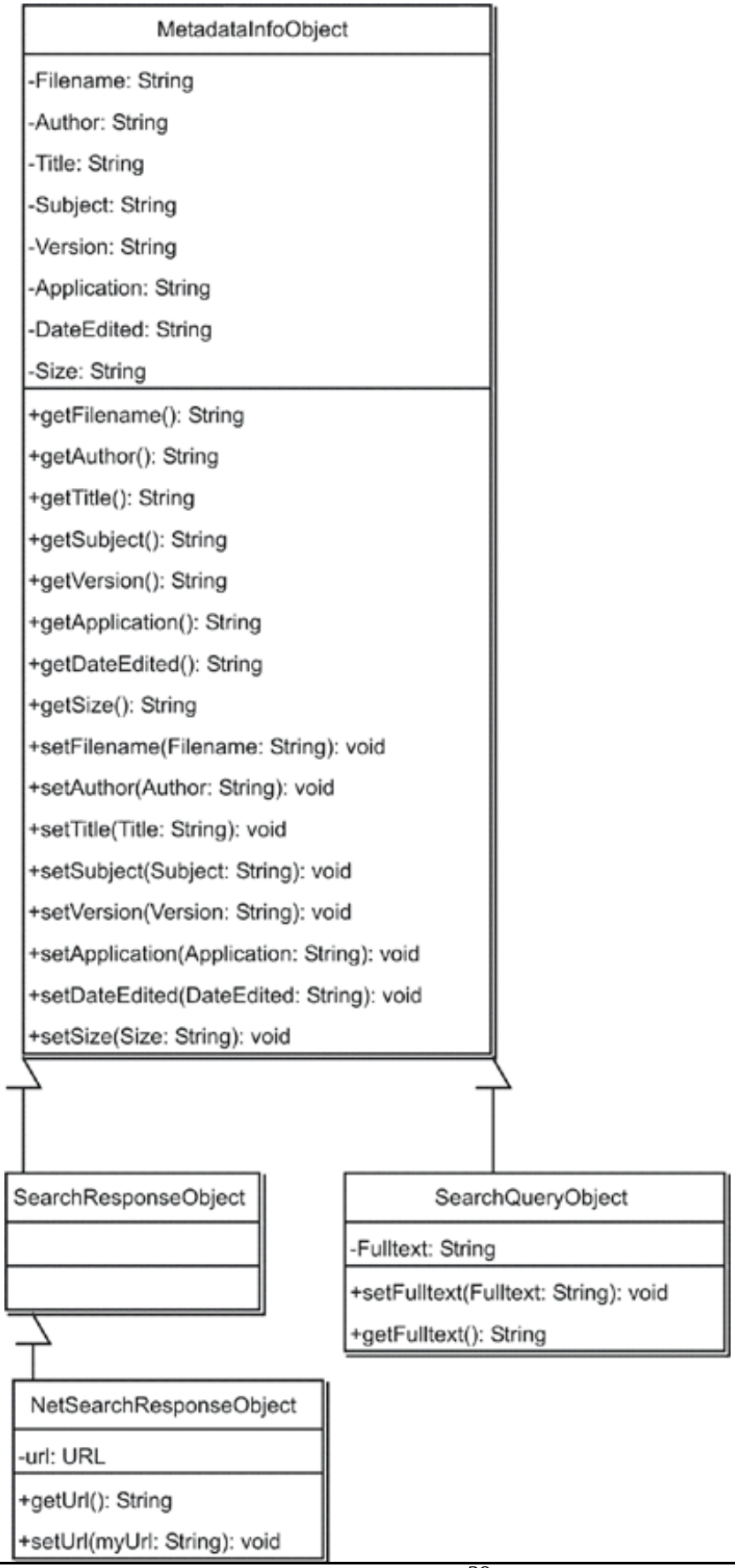
Figure 28.1. class Indexer



class MetaDataInfoObject

This class holds information of a file's metadata.

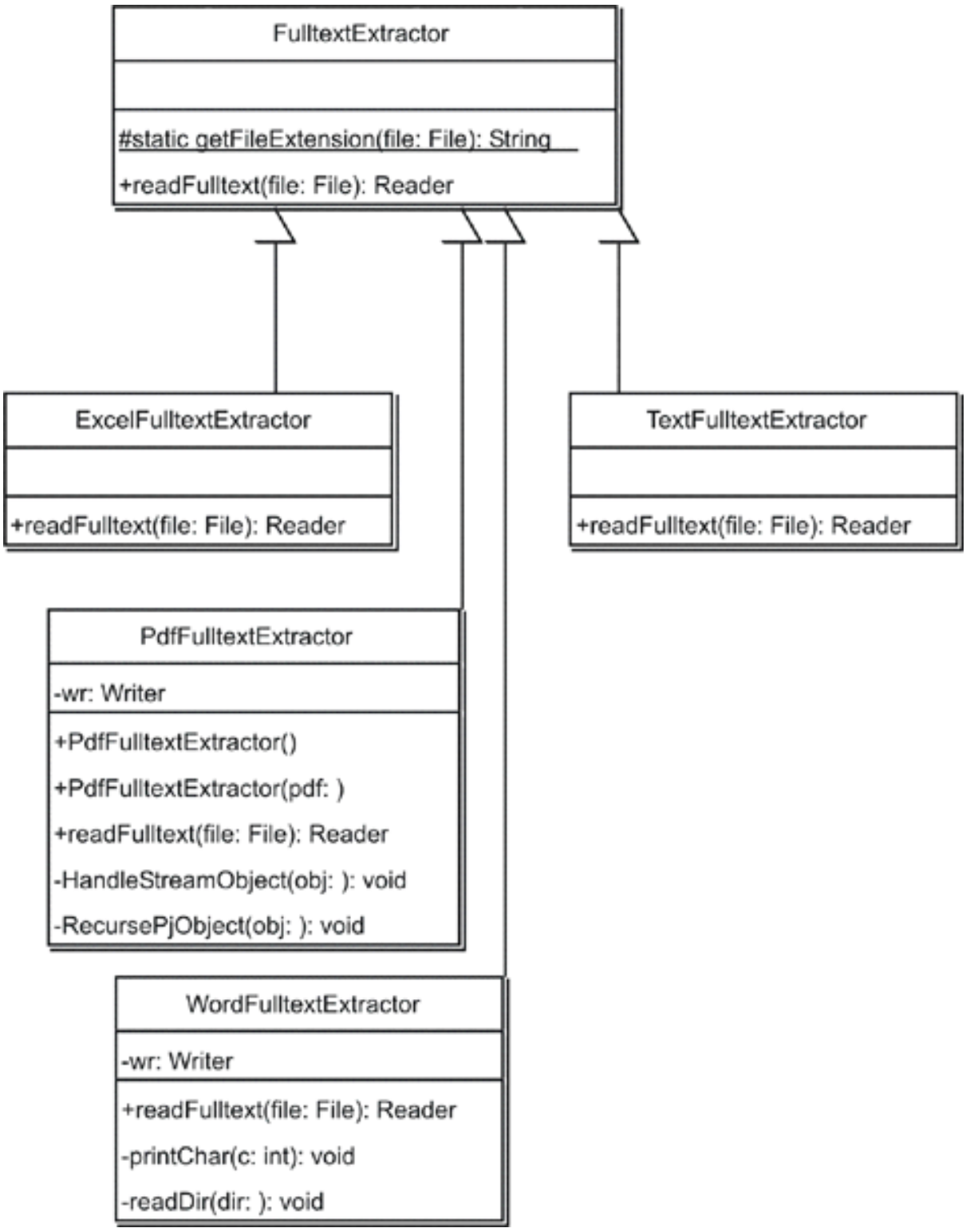
Figure 28.2. class MetaDataInfoObject


MetadataInfoObject
-Filename: String
-Author: String
-Title: String
-Subject: String
-Version: String
-Application: String
-DateEdited: String
-Size: String
+getFilename(): String
+getAuthor(): String
+getTitle(): String
+getSubject(): String
+getVersion(): String
+getApplication(): String
+getDateEdited(): String
+getSize(): String
+setFilename(Filename: String): void
+setAuthor(Author: String): void
+setTitle(Title: String): void
+setSubject(Subject: String): void
+setVersion(Version: String): void
+setApplication(Application: String): void
+setDateEdited(DateEdited: String): void
+setSize(Size: String): void
SearchResponseObject
SearchQueryObject
-Fulltext: String
+setFulltext(Fulltext: String): void
+getFulltext(): String
NetSearchResponseObject
-url: URL
+getUrl(): String
+setUrl(myUrl: String): void

class FulltextExtractor

Base class to Extract a documents fulltext data. Child classes specialize for special file types.

Figure 28.3. class FulltextExtractor



Core Classes

class CoreService

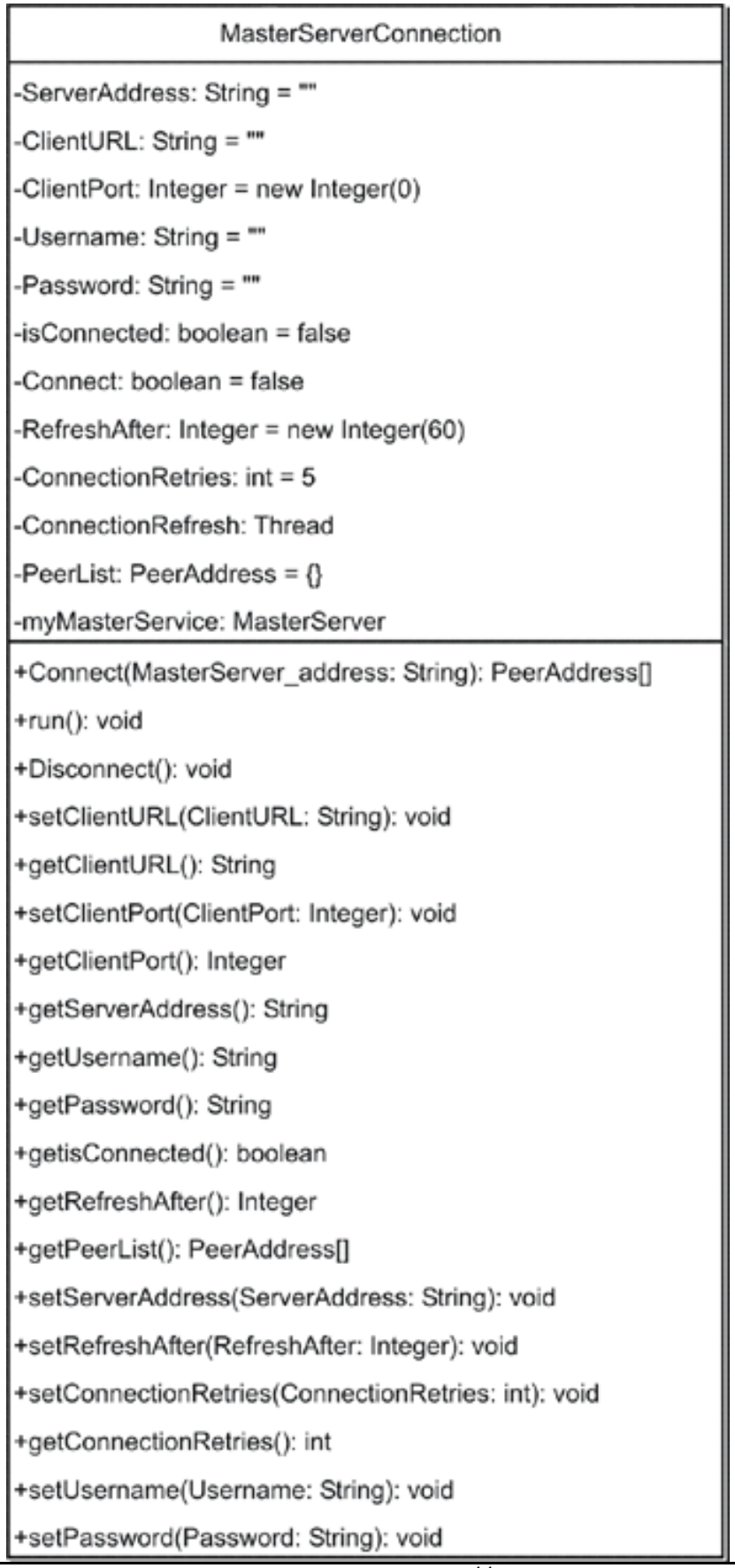
Core and interface of the PaSaMuF client application. This class propagated as a webservice interface to the Gui-Client.

Figure 28.4. class CoreService

class MasterServerConnection

Does connect to a master server and holds information about other peers on the network.

Figure 28.5. class CoreService

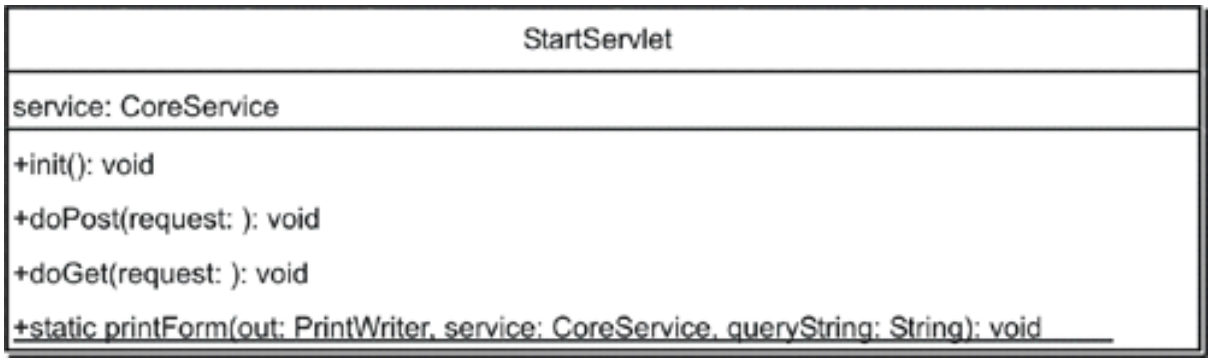

MasterServerConnection
-ServerAddress: String = ""
-ClientURL: String = ""
-ClientPort: Integer = new Integer(0)
-Username: String = ""
-Password: String = ""
-isConnected: boolean = false
-Connect: boolean = false
-RefreshAfter: Integer = new Integer(60)
-ConnectionRetries: int = 5
-ConnectionRefresh: Thread
-PeerList: PeerAddress = {}
-myMasterService: MasterServer
+Connect(MasterServer_address: String): PeerAddress[]
+run(): void
+Disconnect(): void
+setClientURL(ClientURL: String): void
+getClientURL(): String
+setClientPort(ClientPort: Integer): void
+getClientPort(): Integer
+getServerAddress(): String
+getUsername(): String
+getPassword(): String
+getisConnected(): boolean
+getRefreshAfter(): Integer
+getPeerList(): PeerAddress[]
+setServerAddress(ServerAddress: String): void
+setRefreshAfter(RefreshAfter: Integer): void
+setConnectionRetries(ConnectionRetries: int): void
+getConnectionRetries(): int
+setUsername(Username: String): void
+setPassword(Password: String): void

GUI Classes

class StartServlet

This is the simple search servlet.

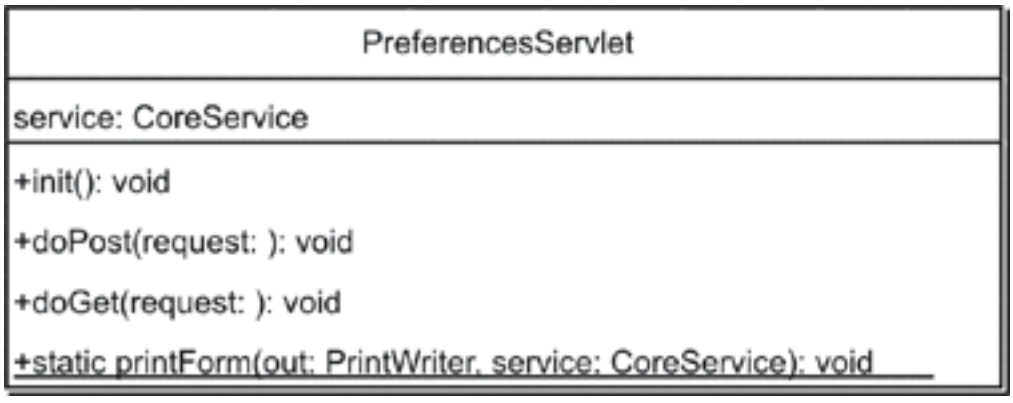
Figure 28.6. class StartServlet



class PreferencesServlet

This servlet lets the user change preferences.

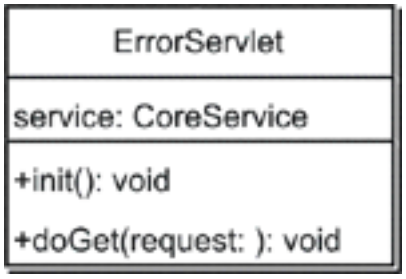
Figure 28.7. class PreferencesServlet



class ErrorServlet

This servlet lets the user control application messages. Mainly used for debugging purposes.

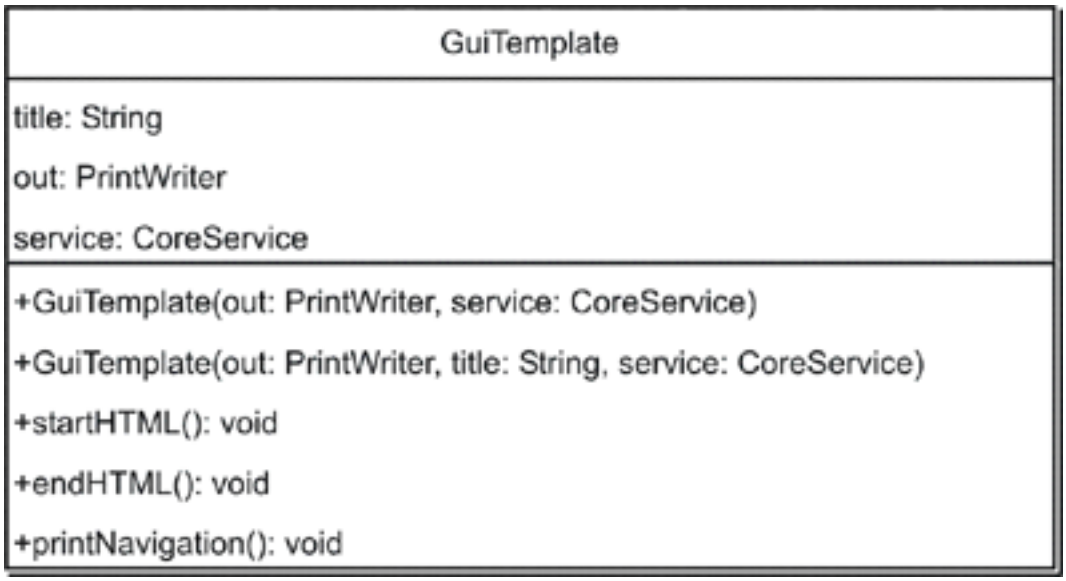
Figure 28.8. class ErrorServlet



class GuiTemplate

This class provides functionality to all servlets. It holds the HTML- Template as well as commonly used methods to build up a Gui- Page.

Figure 28.9. class GuiTemplate

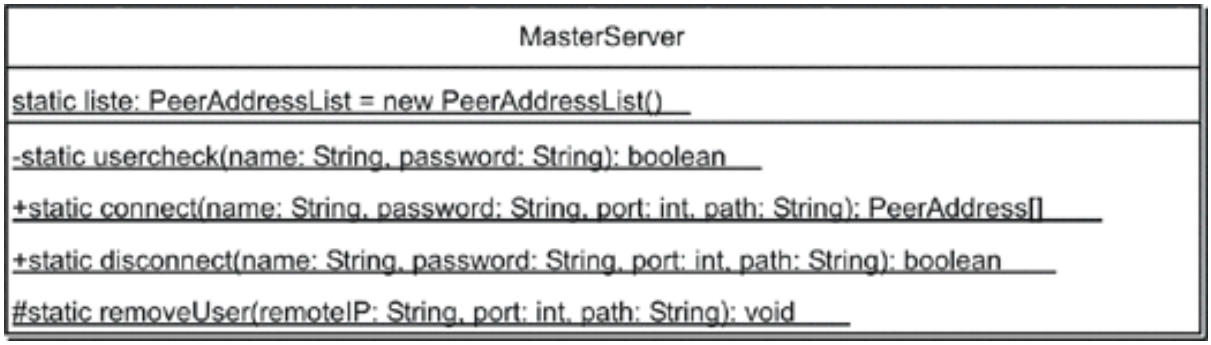


MasterServer Classes

class MasterServer

This class is propagated as a webservice. The clients MasterServerConnection class interacts with this service to authenticate to the PaSaMuF network and get back a list of other connected clients.

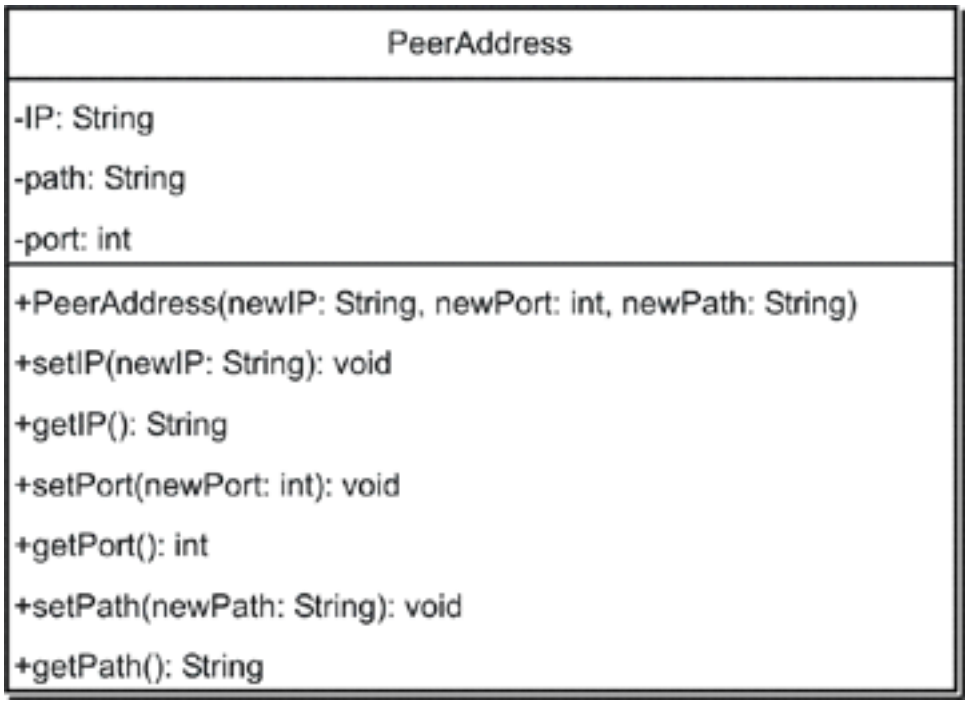
Figure 28.10. class MasterServer



class PeerAddress

This class holds information of one peer necessary for other peers to connect to it.

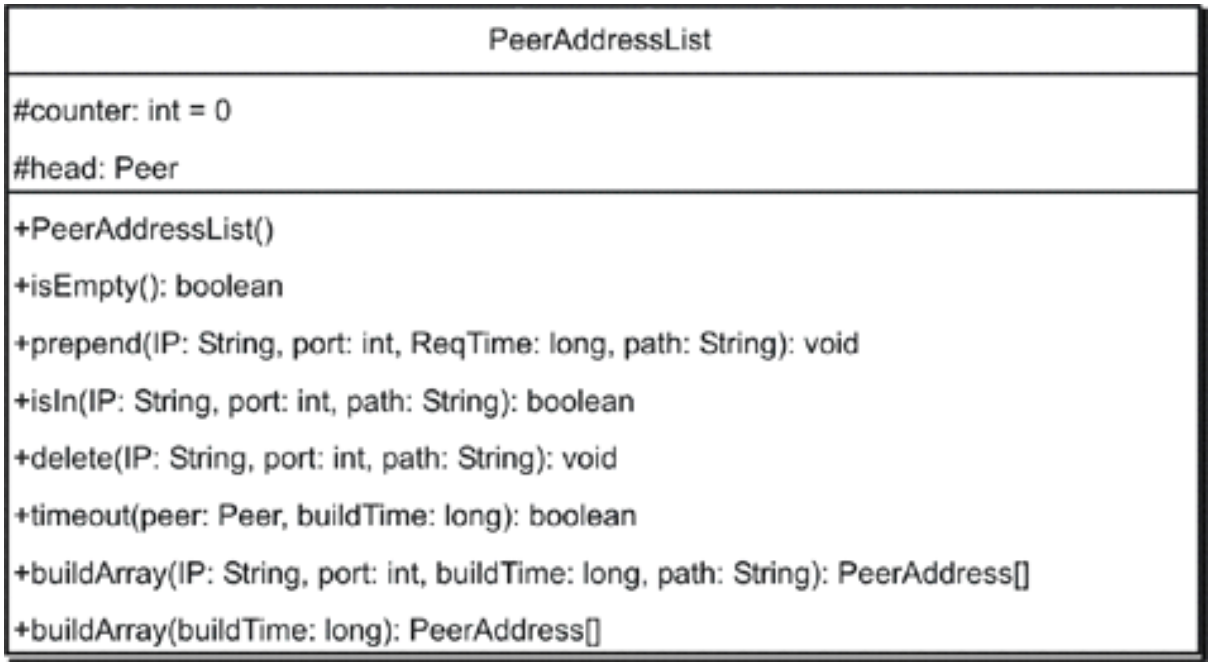
Figure 28.11. class PeerAddress



class PeerAddressList

Manages a list of PeerAddress objects. Does some checks on adding, deleting and getting listelements in order to make sure, the list is valid.

Figure 28.12. class PeerAddressList

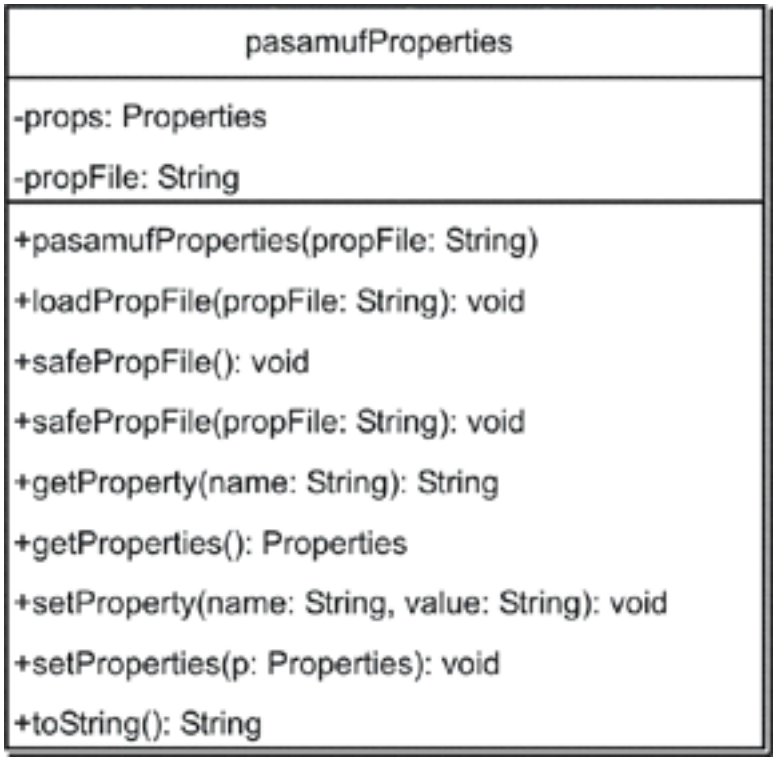


Utility Classes

class pasamufProperties

This class is used by the application core to load and store user preferences from/to a file.

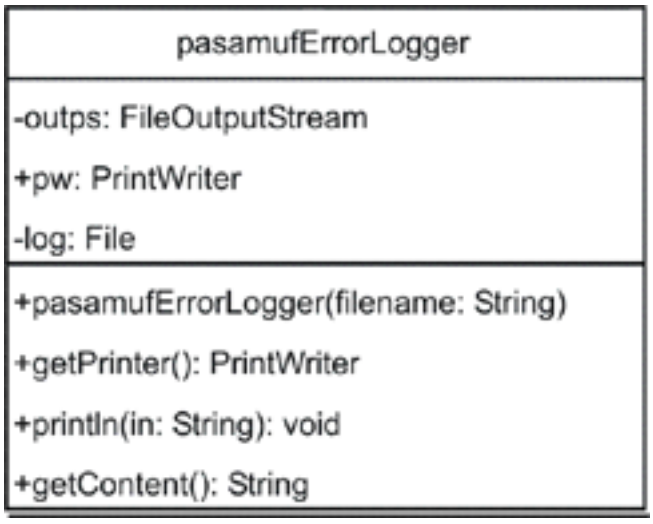
Figure 28.13. class pasamufProperties



class pasamufErrorLogger

This class is used to log errors to a file. This will probably be extended in the future to take several actions to inform developers of errors that occured.

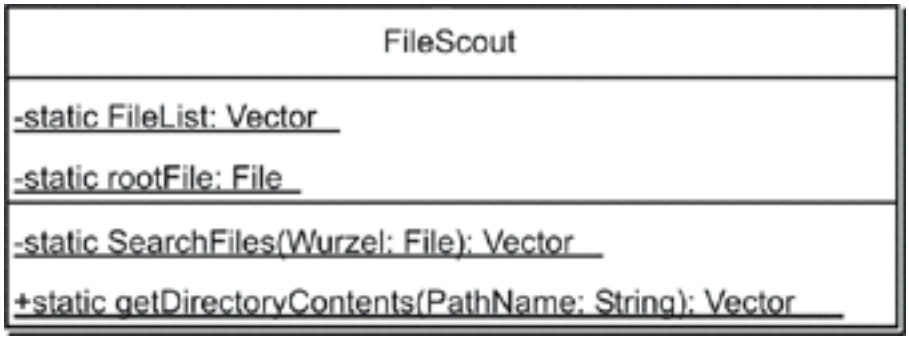
Figure 28.14. class pasamufErrorLogger



class FileScout

Reads a directory recursively and returns all files.

Figure 28.15. class FileScout



Chapter 29. Diary

April

- 12th: Professor Saharoui introduces the idea of a team project.
- 15th: Marc Aßmann begins developing the idea of a document search engine.
- 20th: He is looking for people who like the idea and would like to take part.
- 21st: The project paper is out, containing most of the members already.
- 23rd: The project website (http://www.marc-assmann.de/kdp/) is set up..
- 25th: The project forum (http://www.marc-assmann.de/kdp/phorum) is set up and already in use.
- 27th: The requirements are almost complete (as far as possible up to now).

May

- 3rd: Set up a linux machine with static IP and domain at hpi-project.aip.de
- 15th: MasterServer is implemented and running.
- 24th: Meeting of all project members for discussion of general problems and time planning as well as documentation purposes.

June

- 1st: Properties can be set.
- 5th: Masterserver and its Servlets successfully tested.
- 10th: A Core Webservice has been deployed.
- 15th: Master Server connection almost complete.
- 20th: Marc established a development IRC channel to speed up communication compared to the forum.
- 25th: 2 of 4 GUI Servlets are done.
- 28th: It works: the metafiles and also plane text can be searched through. There are still some small mistakes, and some extractors are missing.

July

- 2nd: GUI is fully functioning.
- 3rd: PaSaMuF works, first steps. The GUI suffers some changes in outward appearance. Slight disagreements about the degree of overridden decisions.
- 5th: PaSaMuF fully functioning.

- 6th: The User Documentation is being completed
- 14th: Marc, being the one best overviewing the whole project, makes some additions and improvements to the project documentation.
- 16th: PaSaMuF being presented in the lecture to Professor Sahraoui and the students .

Chapter 30. The Sub-Projects and Team Members

Abstract

Each member of the project team was asked the following questions:

- What problems did you come across during the project work?
- How did you solve them?
- How and in how far did the members of your project division co-operate?
- Which features would you like to integrate to PaSaMuF if time was not so rare?
- In how far have you been able to enlarge your knowledge during the project work?

Project Management

Marc Aßmann

Marc, as our leading hand, had to have an eye on all PaSaMuF team members. Together with Lars, he established and enforced the use of standard software development methods like the Concurrent Version System

Marc saw slight *problems* only at the beginning of the project, when trying to find the right people for each task. He learnt that working experience is not determinant of a successfull team-member.

He learnt that it is important to make each and every team member clear, that the projects schedule must be met. He will therefore put more attention to this.

Marc dreams of PaSaMuF being a usefull knowlege-management application that can be extended in many way:

- PaSaMuF could offer topic maps to visualize document relationships. These have to be entered by the users themselves or might also be automatically generated via Neuronal Networks like SOM maps.
- PaSaMuF could be integrated into existing environments via file system drivers. Especially a direct integration into office programs could mean a useful and performance-increasing assistance. Furthermore, Marc thinks of the use of a version system in order to provide a variety of obtainable versions, and more intelligent search algorithms that leave the conect of pattern matching behind and match for semantics instead.

Marc also participated in the Master Server division. He studied different possible webservices technologies and decided to work with the AXIS package instead of original Sun Microsystems solutions. He also implemented a small client side webservice test-application for the Master Server in C# (pronounced "C-sharp") on the Microsoft.Net plattform.

During the project work, Marc was able to extremely increase his proficiency in JAVA and has also been reinforced in his experience that looking for an existing solution it is better than immediately trying to solve a problem alone. He also proved once again that intensive as well as extensive communication is essential to the early recognition of problems. Therefore, he set up a project forum right in the beginning and a chat channel by the end of june.

In the hot phase of the project work, Marc is functioning as trouble shooter. When PaSaMuF started working, he

simplified the GUI a little, in agreement with Anja and Roland. As for the documentation, Marc found the most adequate stylesheet, “Windows Help” processing xml into a well readable format.

User Interface (Java, Servlets, Webservices, HTML)

Anja and Roland have been programming the servlets for the graphical user interface between user and application core. They set together, brain stormed, and had long discussions how they could realize their task best. Being only a two person team, they have been able to develop most of the code in direct *co-operation*. The very last details have been added in single work.

Different to static HTML sites, Servlets allow dynamic changes of properties right away. As both of them had to familiarize themselves with Java Servlets first, they had completed about only two of four servlets by the very end of June. However, they managed to complete the missing two servlets until July 3rd.

Tomcat in combination with axis was quite a *problem* to both of them. They found it really hard to get it installed properly. This burden has been overridden with endless trials and errors, as well as Lars' and Marc's help. Another delay was the time they had to wait for the completion and bugfixing of the Core and Webservices in order to be able to program and test the Servlets.

The GUI has been developed primarily based on two means. First HTML-sites, those provide web-communication between the user and the core. Second, Servlets are used for the interface between the user and core. Those Servlets display the HTML-sites in a browser, read accept the user input and hand over the parameters to the core.

Via the graphical user interface, a user can change his preferences or request a search. The search results determined by the core are given to the servlets, which display them in HTML-sites for the user.

Thus to use PaSaMuF (i.e. use the GUI), the user just has to be able to use a browser. The `PreferencesServlet` displaying an HTML-site with input fields, allows him to set the properties needed. When those parameters have been read by the servlet and handed over to the core, the `StartServlet` realizes an easy full text search. This servlet displays an HTML-site with an input field for the search parameters. After the user entered and launched the data, the servlet sends the search parameters to the core. Next, the core sends back the search results to the servlet, where they are displayed as an HTML-site. Each search result is displayed as a link enabling the user to download it.

The `ExtendedSearchServlet` provides an extended search. This servlet is pretty similar to the `StartServlet`, except that with this servlet the user is able to restrict the search by certain parameters, i.e. author, title, date etc. Between searching other peers' documents and own documents, the results differ in that a file on one's own PC is directly linked, whereas a distant file is linked via an access servlet.

To make the programming of the HTML sites easier in the servlets, there is a `GuiTemplate` file that has basic elements of an HTML site, which are build in the servlets above. For the start there is an `InitializationServlet`. This servlet displays an HTML-site that allows the user to choose whether to set properties or to search.

Slight *changes* of the first concept have proceeded in the beginning, when they agreed to program everything in Servlets only, to also generate HTML sites, instead of creating static HTML sites to be accessed by Servlets. However, to the very end, the outward appearance has undergone another radical change.

Anja Nikoleit

Anja did most of the programming. It took her some time installing Tomcat and Axis, as she had to learn handling these first.

She *learnt* the installation of Tomcat, Axis and deploying Webservices using .wsdl files.

Finally, she contributed to the documentation of the GUI.

Roland Brackmann

Roland is the one who was supposed to develop the design of the servlets.

He *learnt* the concepts of HTML, and how to implement Java Servlets.

Application Core (Java, Servlets, Webservices)

The local installation of AXIS und ANT turned out not to be necessary. The essential files have been packed into CVS and the build procedure, i.e. they wrote a batch file that made a deeper understanding of ANT superfluous.

Design issues have been discussed by Lars and Martin. Any further *co-operation* in the core team has been strictly restricted to receiving instructions from Lars, who organized the processes here, and the others trying their best. Each member was supposed to code on his own and check in the files into CVS. However, the use of CVS was new to almost everyone and not always easy in the beginning, when the members preferred sending their files via eMail, which has complicated co-operation in the first time.

Richard Metzler

On first view Richard did not contribute much code to the project, which has been proved true on the second view. But he supported the whole project, by preventing the crew from coding from the start and insisting on writing requirements first before starting to code. Of course, the crew members had many different visions of the project and we had many discussions about the architecture and features of PaSaMuF. The discussions did not help much on writing the requirements, so Richard was the one who first spoke to Jan about the project's requirements and motivated him to write the requirements document.

In fact, the only code Richard wrote was the properties class, but he supported his fellows (and especially Martin) by telling them how to solve various Java problems and saving much time for his team thereby.

Lars Trieloff

Lars acceded a very important *role in the whole project*. He set up a server accommodating concurrent versions of files being developed. and impelled the whole project team to check in their files into CVS and functions as technical advisor for everyone having trouble with Tomcat, Ant, Java and XML

His *part in the sub-project* was the development and coding of the core module webservice and distributing tasks among the members of his sub-project, the core.

Lars would have preferred a more effective co-operation right from the beginning, using CVS more intensively. Unfortunately, he lacked some time, as this project was not the only effort that HPI professors required within this time.

As these problems are more or less of a very subjective kind, Lars found himself able to react only by pushing the others and simply waiting.

Details that have been changed or edited out: It was planned to propagate the search request, but this idea was rejected by the project manager, Marc, for time being too scarce.

Lars was able to *broaden his knowledge*about Simple Object Access Protocol and learnt to convey his own will and drive to the others.

Further ideas whirring in his brain are a trial of the MinTC embedded ServletContainer, to write superior buildfiles, and last but not least, the realization of the documentation

Martin Probst

Martin had some problems setting up the development environment first, but after he got everything running, including the CVS, he was able to contribute to the project, too.

His two major tasks were writing a class to connect the client with the master server and debugging all the code in the core section. He also wrote a utility class and helped with debugging code from other sections

Martin has had some programming experience with C-like languages, so learning the Java syntax was not a big deal to him, but the complex structure of packages, classes, etc. was something he had to find his way through.

He learnt a lot about exceptions in general, remote WebServices and, of course, Java itself. Martin also got some impressions of how group collaboration in big open source projects might work.

Benedikt Meuthrath

Benedikt was responsible for the build.xml, which integrated all application parts for installation.

This file had to be upgraded and improved incessantly and was the component to be completed at the very end. Benedict's part in the project can therefore hardly be temporally ranged in.

ANT was totally new to Benedikt, he had to familiarize with it first in order to be able to create a build procedure as simple as possible for the user.

He *learnt*what a build procedure is, what ANT is, why we need CVS (thanks to Lars), and not to take frequent pressure from the project management too personally.

Moreover, he found out that he had a serious knowledge lack concerning the oragnisation of team software development as such.

Master Server (Java, Servlets, Webservices)

The master server is the central organ of our PaSaMuf FileSharing Tool It is responsible for the peers' Login to and -off of the network.

The physical server has been installed and worked steadily almost from the beginning of the project duration. Servlets implementation and installation on the physical server have been completed and successfully tested on June 5th. Software needed: Tomcat and Axis, both of Apache Jakarta Project. Documentation in JavaDoc followed last.

The master server is based on an Apache Tomcat Server version 4.0.3 as well as the Apache Axis Servlet. The Java classes needed have to be copied into the AXIS directory before the WebService is deployed with the wsdd-file.

On an incoming request from a client, the Master Server first looks into the "log_dat.txt", if username and password are correct. Depending on the request, it will then connect or disconnect

See the petrinet for these functions. The array that is sent back must not contain the own address.

The "log_dat.txt" is structured as follows:

- first line contains the file's number of entries
- following lines contain one entry per user, each entry consisting of name; password;

This file is to be maintained by the administrator.

The Master Server division thought and discussed about problems and implemented everything unitedly. Alex

worked out a rough concept, which has then been discussed and refined in the group. Particulars have been implemented both in *group and single work*. For every next group intern meeting goals have been set for each member

They did not see themselves confronted to any serious problems. Some *slight obstructions* occurred during code implementation due to slightly mistaken algorithms. These have been successfully debugged. The only *problem* emerged when trying to get the IP of a requesting client with the servlet having been generated with Axis WSSD files This problem has been solved within a very short amount of time by project manager Marc Aßmann studying the Axis architecture.

One thing that had to be *changed* was the planned cyclic ping to all clients, as this was no sensible solution with changing IPs at T-Online Instead, a cyclic reconnect by the clients was considered to be the better solution.

The Master Server was the first fully coded and functioning part of the project, which meant a good basis to test the other parts while developing Comments and detailed documentation have been added afterwards by Johannes.

For *further developments* on PaSaMuF, Alex would prefer an administration tool for this file, handling MD5-encrypted passwords. Not even the administrator would be aware of the passwords

A little peak in trouble shooting was getting the user IP, which was finally realized using AXIS methods.

Alex Saar

Alex acceded the role of the sub-project manager which, except from coding, meant time planning, fixing target dates and keep up communication with the Core sub-project. Moreover, he installed a physical server with a fixed IP as a basis for the Master Server

Alex *learnt* how to install a physical server as well as Tomcat and Axis, and deploying webservices via WSDD-files for Axis.

Due to the cyclic reconnects of the clients, the password is unencryptedly transferred every minute. One of the very first *ideas for improvements* to PaSaMuF has been the correction of this unacceptable security lack, which, however, turned out to be much more complex than presumed and therefore could hardly be realized within the time given. Possible solutions might be a hashing algorithm based on MD5, for example. Another idea is the data transfer via SSL, which would consume far too much time implementing it oneself on the one hand. On the other hand, a ready-built SSL key would be extremely expensive

Jan Hartmann

Jan especially, whose experiences with JAVA did not exceed the training in Concepts of Algorithms, was able to learn JAVA quite well during the project work For the first time, he usefully deployed data structures like lists and got in contact with servlet architecture.

Jan did not only plan and code in the Master Server division, he also set up the requirements specification together with Marc for further use in the project documentation.

Johannes Wust

Johannes was especially responsible for post-commentaries in JAVADOC and Code improvement in the Master Server division. He learnt some programming in JAVA and how to work in group efficiently.

Indexing Module

Co-operation within the sub-project? The indexing people have been working mainly alone, with the exception of one afternoon at HPI. They have been communicating at the HPI as well as via eMail and, in the hot phase also via IRC, which allowed non-relayed communication and co-operation at the computer, without the neces-

sity of being physically together.

Changes? A self-developed solution for saving and indexing meta data (Alex) and full text (Florian and Ole) has been planned. However, this would have become quite time-consuming and probably very slow. Therefore, they used the Lucene sub-project from Apache Jakarta (http://jakarta.apache.org), which has fully satisfied all their needs and provides more functionality than could have been developed by themselves.

The real search engine is Lucene, by Apache Jakarta [http://jakarta.apache.org/], which has got a the section called “Lucene Query Syntax”variable syntax. This is manually usable in our SimpleSearch as well as in the MyDocuments search function. The Indexer (indexing/Indexer.java) generates the search automatically. The fields we save in Lucene for each file are:

- "author"
- "application"
- "title"
- "subject"
- "filename"
- "text"
- "size"
- "date_edited"
- "version"

In fact, both SimpleSearch and ExtendedSearch are really searching only in the fields,"author", "application", "title", "subject", "filename" and "text".

SimpleSearch links the strings given with a logical OR, i.e. either string having been found in either field returns a positive search result

ExtendedSearch links all the given strings with a logical AND. For example, if the user types a query like

author = Frank, application = Excel, title = test, subject = , filename = xls, text = calculation of, Lucene gets the following query string: author:"Frank" AND application:"Excel" AND title:"test" AND filename:"xls" AND text:"calculation of"

Obviously, it is formatted `field to search in`: "`string to be searched`". Subject is left out as nothing has been typed in here. The logical AND enforces all given entries to be found for a positive search result.

Alex Klimetschek

Alex K. was responsible for the sub modules of the Indexing module, which consist of classes extracting meta files and plane text from files.

Beside small *problems* not worth mentioning mainly transferring extracted texts to the Lucene Indexer caused some complications. First, a stream in the JAVA API was missing, which is able to hold a larger amount of data in the main memory. Also trials with PipedReader/-Writer have failed.

The extraction from the essential document types specified by the requirements (PDF, Word, Excel) needed the help of other projects. The project for MS Office™ files is POI, also part of Apache Jakarta (http://jakarta.apache.org/poi). Unfortunately, the internal MS Word format is currently not supported by POI.

The project extracting from Adobe PDF files is PJ [http://www.etymon.com/pj]. This one is badly documented, so Alex had to study the Adobe PDF Reference very intensively.

Alex could manage the data transfer problem with the help of `CircularCharBuffer` (from http://ostermiller.org/utils/#download), which he simply integrated into the existing JAVA package. It allows writing as well as reading and is therefore perfectly suitable for the purpose in question For text extraction from MS Word, Alex had to read through the file format reference by Microsoft and write corresponding routines himself

Alex was able to get *more intensively into* JAVA programming and, in this context, reading Javadocs. Moreover, he learnt about the structure of some file formats, among these are most notably Adobe PDF and MS Word, but also the MS Office Format OLE 2.

For *future extensions* to the indexing module, Alex thinks of developing further meta data and plane text extractors, especially for OpenOffice files (http://www.openoffice.org), but also audio and video files like MP3, AVI etc.

Florian Brodersen

What problems did you come across during the project work?

At first, it wasn't quite clear how the indexing stuff would be implemented. The group thought about doing it by themselves but the estimated amount of time they would have spent creating their own solution would have spoiled the project schedule. One of the indexing group soon found out that the Apache Software Foundation had already worked out a method for indexing documents. So they happily decided to use it.

How did you solve them?

As mentioned above, they solved the problem by doing some research in the internet.

How and in how far did the members of your project division co-operate?

The members of this project division did co-operate pretty well even though they didn't do much in the first few weeks. But later on, they met regularly in the irc to discuss the current development and to communicate directly within the group.

Which features would you like to integrate to PaSaMuF if time was not so rare?

Flo would like to see Lucene doing regular expression searchs so that one could do more powerfull lookups. This is actually a task which is on the current "roadmap" for the lucene project but nobody has yet volunteered for this.

In how far have you been able to enlarge your knowledge during the project work?

As a matter of fact, Flo has not worked with Java before. This was not such a big problem, as the learning curve is pretty steep. Nevertheless, he gained some expertise in that area.

Ole Weidner

Beside some small efforts in the indexing team, Ole took responsibility for the preparation of the project presentation.

Documentation

Thomas Wendlandt

The documentation of the development process as well as the end user documentation have been created in XML using The Docbook DTD

Lars was the one who proposed using Docbook for documentation and introduced me into that technique.

During my work for the PaSaMuF project I learnt to write documents in the Docbook format, however still not being able to use to its full extend, as the author has to know quite a lot of tags for efficient use. I appreciate this documentation technique as very useful for large and structured documentation projects like this. Unfortunately, it took me quite a piece of our scarce time to get into it.

Testing

PaSaMuF has undergone its testing in the last week, as soon as it started to work. All members tested the basic functionalities as meant in regular use.

Conclusion

Almost everyone invested a considerable amount of time into familiarization with development methods, e.g. tools like CVS. These consumed more or less time in the beginning, when trying out their essential behaviour, but, if sensibly used, save much more time in long term use. Unfortunately, as for the short time of this project, we had some overhead here. However, getting into these techniques and organizing teamwork seemed to be Professor Sahraoui's primary intention for this project.

Part III. User Documentation

Chapter 31. Installation

Installation Prequisites

In order to install und run PaSaMuF, you will need to have following software installed on your computer.

- **Operating System.** PaSaMuF needs an operating system for which a Java Software Development Kit is available. It has been tested under Windows95-WindowsXP and Linux.
- **WWW-Browser.** Most systems will already have a web-browser installed, otherwise you will need to install one. PaSaMuF will work with nearly every Web-Browser available, even Lynx.
- **Java 1.4 Software Development Kit.** PaSaMuF needs a Java Software Development Kit for deploying the included soap components. Additionally the Java Software Development Kit includes a Java Runtime Environment, which you will need for running java-applications like PaSaMuF. The Java Software Development Kit 1.4 can be downloaded from Sun's Java Site [http://java.sun.com/j2se/1.4/download.html].
- **Apache Tomcat 4.0.4.** Because PaSaMuF uses servlets for the user interaction and for the webservices communication, it will need a servlet engine like Apache Jakarta Tomcat. PaSaMuF has been tested with Apache Jakarta Tomcat 4.0.4 which you can download from the Jakarta Distribution's Site [http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.4/]. Make sure your "CATALINA_HOME" environment variable is properly set.

When you have all these prequisites installed, you can continue the installation proccess.

Compiling and Installing PaSaMuF

Compiling and installing PaSaMuF is only one step, so we do not provide binary distributions. You can get the latest sources either by downloading from cvs[2] or by downloading a nightly snapshot from the source tree at http://trieloff.dyndns.org/webcvs/JCVSlet/zip/pasamuf/pasamuf/pasamuf.zip [http://trieloff.dyndns.org/webcvs/JCVSlet/zip/pasamuf/pasamuf/pasamuf.zip].

After you have extracted the package, you can start compiling and installing by moving to the extracted directory and typing at the command line:**build** (for Windows Users) or **sh build.sh** (for Linux Users).

Wait for the output `BUILD COMPLETED`. Now PaSaMuF is installed.

Step by Step Installation of PaSaMuF client

1. Open a text-shell and change to the root of this distribution.

2. Edit the file "config.properties" and set the correct values for the following fields:

- client.password
- client.username (both provided byMasterServer Admin)
- masterserver.host (MasterServer Hostname)
- masterserver.port

2 Using $CVSROOT=:pserver:trieloff.dyndns.org/usr/local/cvsroot with username `anoncvs` and password `anoncvs`.

- masterserver.dir
- core.shareddir (your personal dir of your documents to share)

(the ones explaned in brackets are most likely to be changed).

3. Enter

- Windows: "build prepareaxis"
- Linux: "./build.sh prepareaxis"

4. Start tomcat by executing

- Windows: %CATALINA_HOME%/bin/startup.bat
- important to do so, not using a shortcut!
- Linux: %CATALINA_HOME%/bin/startup.sh

5. Enter

- Windows: "build deploycore"
- Linux: "./build.sh deploycore"

6. Restart Tomcat

7. Open a browser with the address "http://localhost:8080/axis" to test the install.

Step by Step Installation of PaSaMuF MasterServer

1. same as above.

2. Edit the file "users.txt" and enter your list of users. Enter a line for each user, starting from the second line. A user-password combination is enter the following way: username;password;

There is one default entry to show you how it works. You should remove this user now.

3. Enter

- Windows: "build preparemaster"
- Linux: "./build.sh preparemaster"

4. Start tomcat by executing

- Windows: %CATALINA_HOME%/bin/startup.bat *It is very important to do so, not using a shortcut!*
- Linux: %CATALINA_HOME%/bin/startup.sh

5. Enter

- Windows: "build deploymaster"
- Linux: "./build.sh deploymaster"

6. Restart Tomcat

7. Open a browser with the address "http://[your domain]:[your port]/axis/services/MasterServer?wsdl" to test the install. You should see an WSDL XML-file describing the MasterServer service.

Chapter 32. Running PaSaMuF

The Structure

PaSaMuFs Graphical User Interface is based on a web browser. Each of its windows provides a navigation bar at its top and its bottom, where another window might be activated. The name of the active window is always highlighted in this navigation bar.

- Search
- Extended Search
- My Documents
- Preferences
- Application Log
- Credits

Changing the Preferences

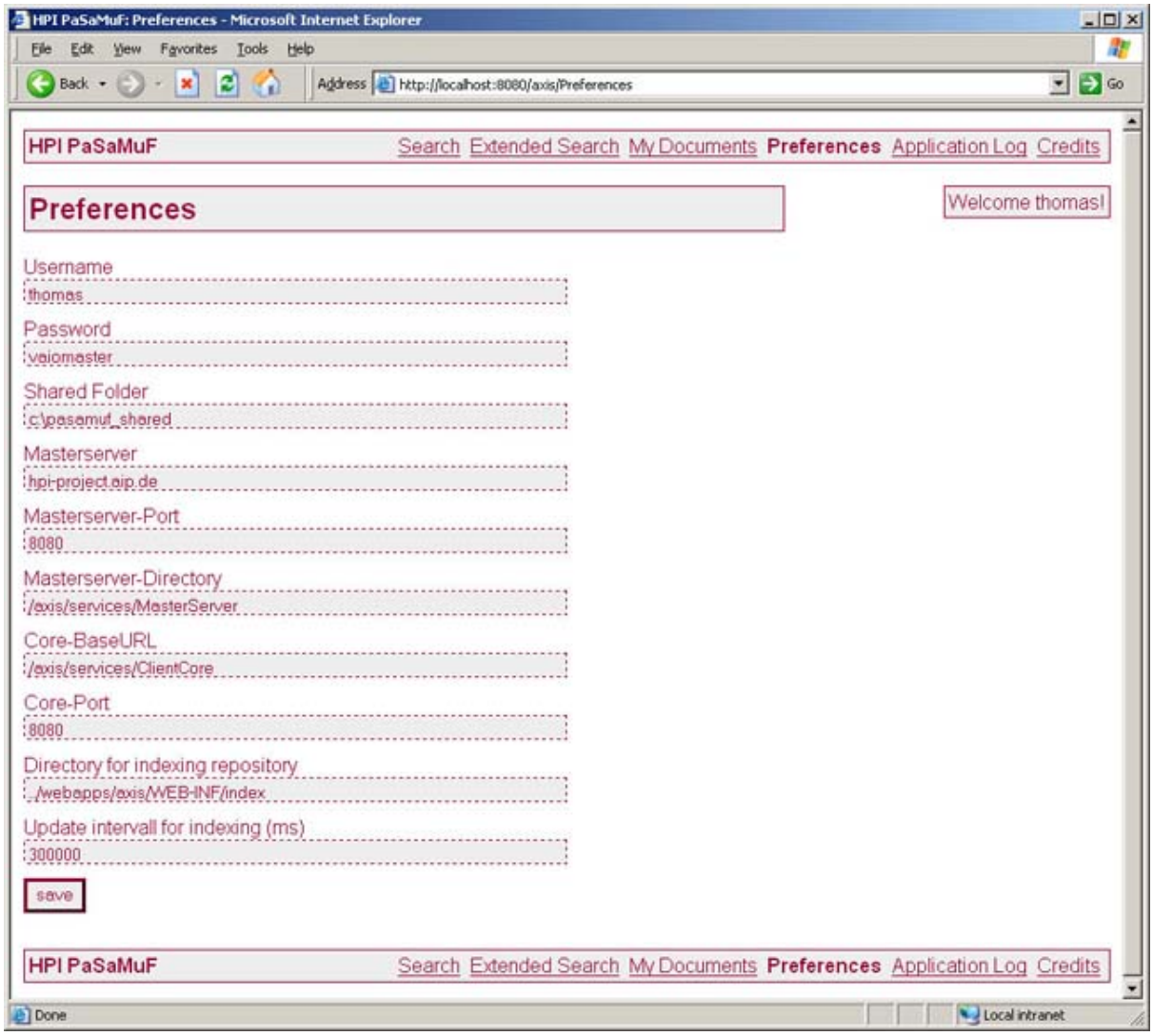
Before you can search documents, or let other users search your documents, you have to set the parameters needed in the `Preferences` window. You have to type in your username and the corresponding password you received from the administrator, currently Alex Saar.

In `Shared Folder`you have to specify the folder containing the files you would like to share with other peers.

The Master Server and Core settings are predefined by the installation and should be changed only if installation paths have changed.

The update interval for indexing is set to 5 minutes by default. A smaller interval is sensible only in times of extreme document traffic.

Figure 32.1. An example for preferences

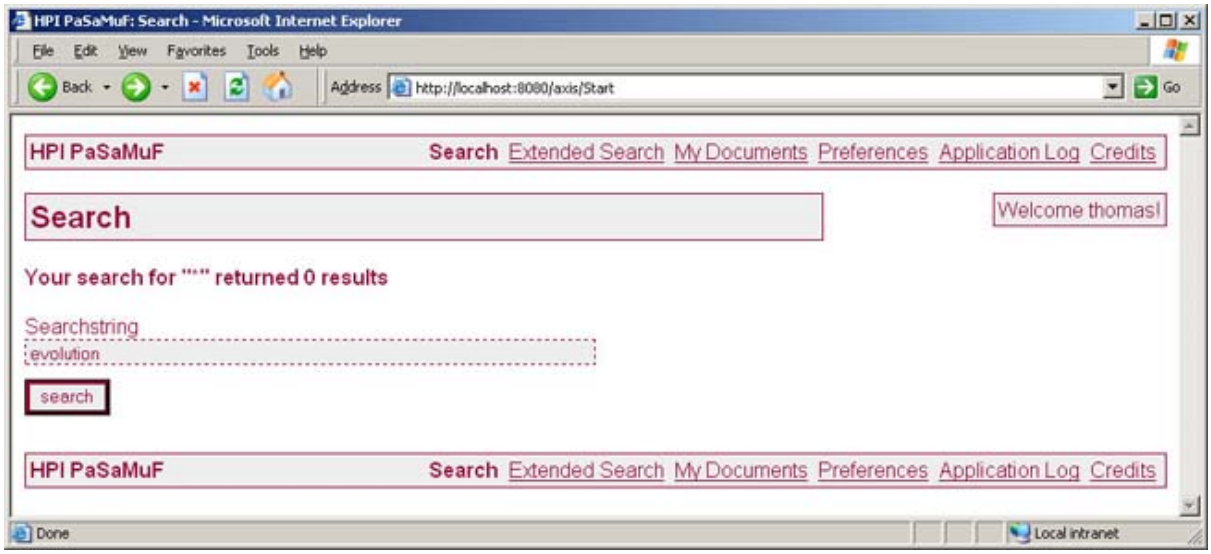


Searching

Simple Search

The simple search returns all the files containing the given string in any part of the file, i.e. either in the file name, in any meta data or in the full text. If you type several strings, PaSaMuF will return all files containing any of the strings given. You could generally speak of a logical OR linkage, concerning the given strings as well as the categories they are searched in.

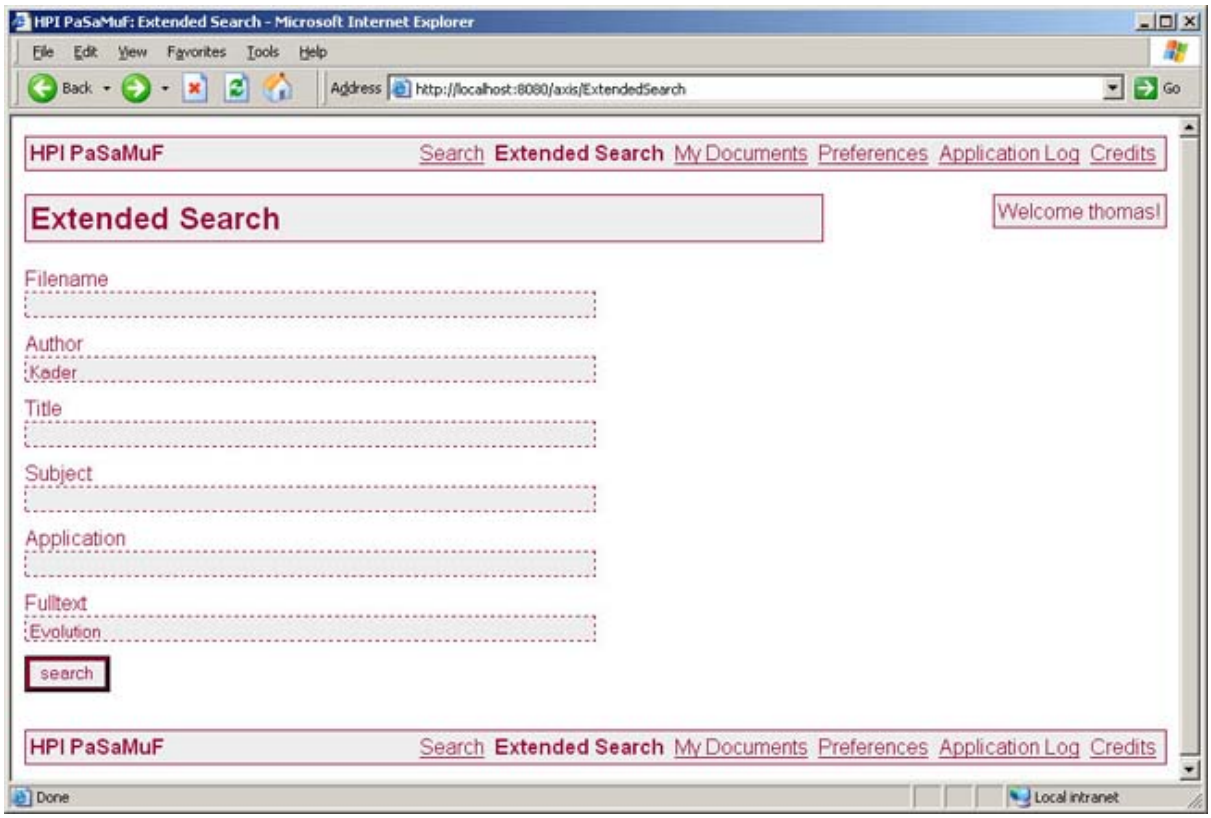
Figure 32.2. Simple Search



Extended Search

The extended search lets you specify, in which category: file, meta data or full text, the given string shall be searched. The strings you type in here are searched in a logical AND linkage, i.e. every single entry must be found as typed, in the datum you specified. For example, an extended search with the given strings *title*= “TestFile” and `Full Text`=“calculation of” will only return a file containingtitled ontaining BOTH the string “TestFile” in the title meta datum AND “calculation of” in the full text part.

Figure 32.3. Extended Search



For both simple and extended search a table shows the search results, each row containing the file name, the meta data `Author, Title, Subject, Version, Application` as well as the date the file has been edited and its `size`.

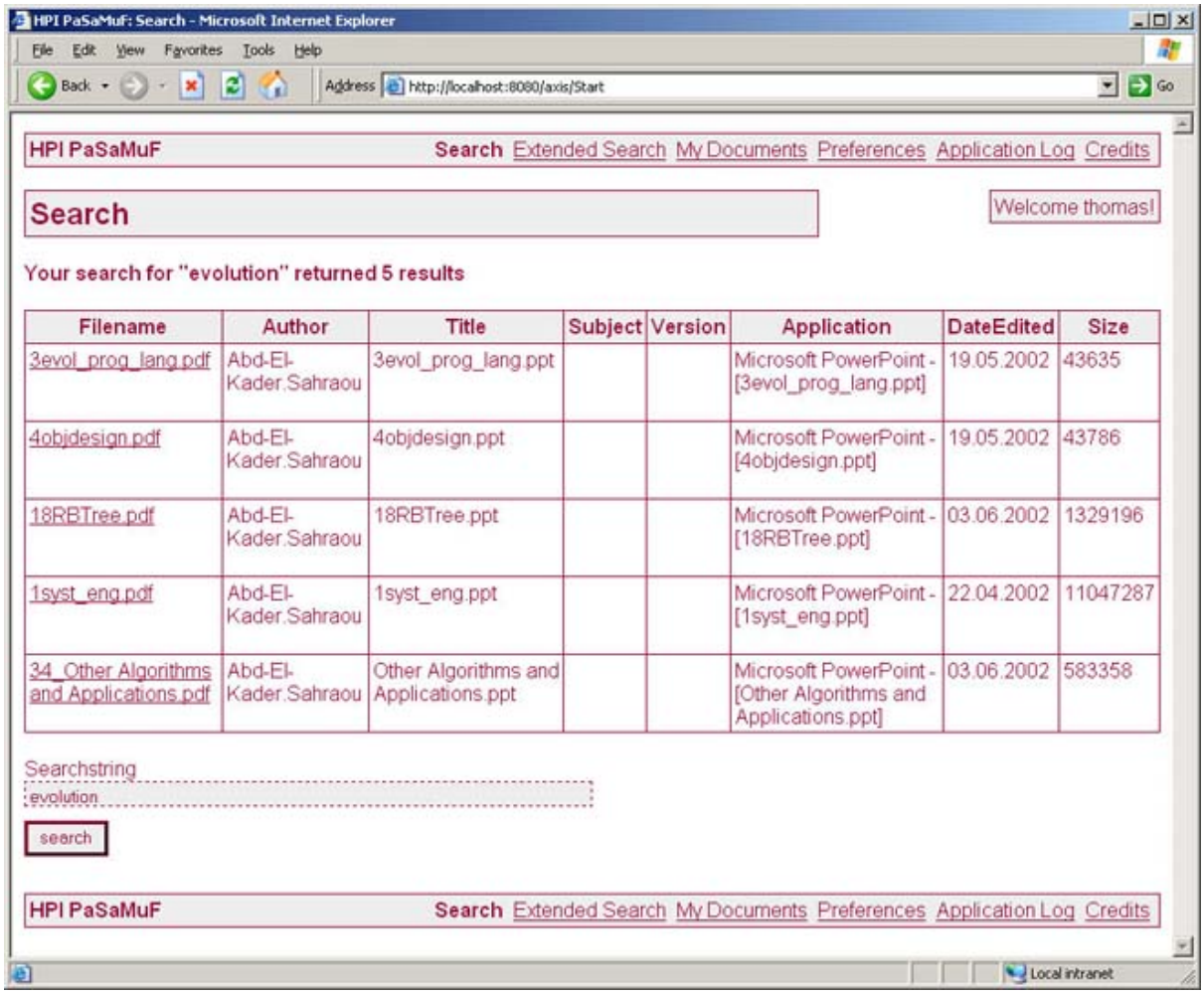
My Documents

PaSaMuF allows you not only to search documents from other peers but also your own Shared Documents. Choose the entry `My Documents` in the navigation bar and just use it like `Simple Search`.

Downloading Files

The graphic shows a list of files that PaSaMuF found at other peers, providing a link to the file and the meta data that could be extracted.

Figure 32.4. An example of search results



Just klick the link to the file (i.e. the file name) in the leftmost column and you can download it.

Lucene Query Syntax

A query is broken up into terms and operators. There are two types of terms: Single Terms and Phrases.

a. A Single Term is a single word such as "test" or "hello".

b. A Phrase is a group of words surrounded by double quotes such as "hello dolly".

Multiple terms can be combined together with Boolean operators to form a more complex query (see below).

Note: The analyzer used to create the index will be used on the terms and phrases in the query string. So it is important to choose an analyzer that will not interfere with the terms used in the query string.

Fields

Lucene supports fielded data. When performing a search you can either specify a field, or use the default field. The field names and default field is implementation specific.

You can search any field by typing the field name followed by a colon ":" and then the term you are looking for.

As an example, let's assume a Lucene index contains two fields, title and text and text is the default field. If you want to find the document entitled "The Right Way" which contains the text "don't go this way", you can enter:

```
title:"The Right Way" AND text:go
```

or

```
title:"Do it right" AND right
```

Since text is the default field, the field indicator is not required.

Note: The field is only valid for the term that it directly precedes, so the query

```
title:Do it right
```

Will only find "Do" in the title field. It will find "it" and "right" in the default field (in this case the text field).

Term Modifiers

Lucene supports modifying query terms to provide a wide range of searching options.

Wildcard Searches

Lucene supports single and multiple character wildcard searches.

To perform a single character wildcard search use the "?" symbol.

To perform a multiple character wildcard search use the "*" symbol.

The single character wildcard search looks for terms that match that with the single character replaced. For example, to search for "text" or "test" you can use the search:

```
te?t
```

Multiple character wildcard searches looks for 0 or more characters. For example, to search for test, tests or tester, you can use the search:

test*

You can also use the wildcard searches in the middle of a term.

te*t

Note: You cannot use a * or ? symbol as the first character of a search.

Fuzzy Searches

Lucene supports fuzzy searches based on the Levenshtein Distance, or Edit Distance algorithm. To do a fuzzy search use the tilde, "~", symbol at the end of a Single word Term. For example to search for a term similar in spelling to "roam" use the fuzzy search:

roam~

This search will find terms like foam and roams

Note:Terms found by the fuzzy search will automatically get a boost factor of 0.2

Proximity Searches

Lucene supports finding words are a within a specific distance away. To do a proximity search use the tilde, "~", symbol at the end of a Phrase. For example to search for a "apache" and "jakarta" within 10 words of each other in a document use the search:

"jakarta apache"~10

Boosting a Term

Lucene provides the relevance level of matching documents based on the terms found. To boost a term use the caret, "^", symbol with a boost factor (a number) at the end of the term you are searching. The higher the boost factor, the more relevant the term will be.

Boosting allows you to control the relevance of a document by boosting its term. For example, if you are searching for

jakarta apache

and you want the term "jakarta" to be more relevant boost it using the ^ symbol along with the boost factor next to the term. You would type:

jakarta^4 apache

This will make documents with the term jakarta appear more relevant. You can also boost Phrase Terms as in the example:

"jakarta apache"^4 "jakarta lucene"

By default, the boost factor is 1. Although, the boost factor must be positive, it can be less than 1 (i.e. .2)

Boolean operators

Boolean operators allow terms to be combined through logic operators. Lucene supports AND, "+", OR, NOT and "-" as Boolean operators(Note: Boolean operators must be ALL CAPS).

OR

The OR operator is the default conjunction operator. This means that if there is no Boolean operator between two terms, the OR operator is used. The OR operator links two terms and finds a matching document if either of the terms exist in a document. This is equivalent to a union using sets. The symbol || can be used in place of the word OR.

To search for documents that contain either "jakarta apache" or just "jakarta" use the query:

```
"jakarta apache" jakarta
```

or

```
"jakarta apache" OR jakarta
```

AND

The AND operator matches documents where both terms exist anywhere in the text of a single document. This is equivalent to an intersection using sets. The symbol && can be used in place of the word AND.

To search for documents that contain "jakarta apache" and "jakarta lucene" use the query:

```
"jakarta apache" AND "jakarta lucene"
```

+

The "+" or required operator requires that the term after the "+" symbol exist somewhere in a field of a single document.

To search for documents that must contain "jakarta" and may contain "lucene" use the query:

```
+jakarta apache
```

NOT

The NOT operator excludes documents that contain the term after NOT. This is equivalent to a difference using sets. The symbol ! can be used in place of the word NOT.

To search for documents that contain "jakarta apache" but not "jakarta lucene" use the query:

```
"jakarta apache" NOT "jakarta lucene"
```

Note: The NOT operator cannot be used with just one term. For example, the following search will return no results:

```
NOT "jakarta apache"
```

-

The "-" or prohibit operator excludes documents that contain the term after the "-" symbol.

To search for documents that contain "jakarta apache" but not "jakarta lucene" use the query:

```
"jakarta apache" -"jakarta lucene"
```

Grouping

Lucene supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

To search for either "jakarta" or "apache" and "website" use the query:

```
(jakarta OR apache) AND website
```

This eliminates any confusion and makes sure you that website must exist and either term jakarta or apache may exist.

Escaping Special Characters

Lucene supports escaping special characters that are part of the query syntax. The current list special characters are

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

To escape these character use the \ before the character. For example to search for (1+1):2 use the query:

```
\(1\+1\)\:2
```

Appendix A. Contributing to this document

This docbook chapter was initially created by Lars Trieloff to structure his thoughts.

The markup in this chapter orientates among a discussion in the Docbook-Mailinglist by Megan Golding and Michael Smith. You can find this thread [http://lists.oasis-open.org/archives/docbook/200106/msg00086.html] at the Docbook mailinglist archives.

If you would like to contribute to this document, write your part of text as plain-text (bad), html (better) or docbook (best). Then send it to Thomas Wendlandt [mailto: t_wendlandt@web.de], who will manage the documentation.

If you want to write docbook, and have questions, please see the docbook-editing-guidelines in the next section or write a mail to Lars Trieloff [mailto:lars@trieloff.net].

Docbook editing guidelines

For reasons of easy inserting into this book, please send only valid docbook chapters, sections, formalparas, prefaces or appendixes. In the case that you are not submitting a formalpara, please add an according chapterinfo, sectioninfo, prefaceinfo or appendixinfo with information about the author this includes your firstname and your surname.

If you are editing a part of the document, someone else has written please set it's revisionflag attribute to “changed”. This makes it easier to find new content.

Appendix B. Glossary

This is the glossary of this documentation. Words important for understanding the principles of this software are described here.

Glossary

A

Ant

Apache Ant is a Java-based build tool. In theory, it is kind of Make, but without Make's wrinkles. Instead of a model where it is extended with shell-based commands, Ant is extended using Java classes. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by an object that implements a particular Task interface.

AXIS

Apache AXIS is an implementation of the SOAP ("Simple Object Access Protocol") submission to W3C. This project is a follow-on to the Apache SOAP project,essentially a SOAP engine - a framework for constructing SOAP processors such as clients, servers, gateways, etc. The current version of Axis is written in Java, but a C++ implementation of the client side of Axis is being developed. It also includes: a simple stand-alone server, a server which plugs into servlet engines such as Tomcat, extensive support for the Web Service Description Language (WSDL), emitter tooling that generates Java classes from WSDL. some sample programs, and a tool for monitoring TCP/IP packets.

C

Concurrent Version System

CVS maintains a history of a source tree, in terms of a series of changes. It stamps each change with the time it was made and the user name of the person who made it. Usually, the person provides a bit of text describing why they made the change as well. Given that information, CVS can help developers find out, who made a given change? When and why did they make it? What other changes did they make at the same time?

D

The Docbook DTD

DocBook is an XML/SGML vocabulary particularly well suited to books and papers about computer hardware and software (though it is by no means limited to these applications).

Document Type Definition

The purpose of a Document Type Definition is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. A DTD can be declared inline in an XML document, or as an external reference

H

The Hasso Plattner Institute

The HPI was founded in the year 1999 by Hasso Plattner, founder of SAP, with the intention to educate software engineers as needed in real economy.

For the first two years, the HPI headquarters was situated in the LBS Building at Potsdam Luftschiffhafen. Lectures had been read at the University department at New Palace. The 3rd generation of HPI students already had the pleasure to have their very first lectures at the new HPI buildings at Potsdam Griebnitzsee. The HPI together with the Information Technology Faculty of Potsdam University had moved from the west to the east of the city, right to the borders of Berlin.

The HPI Mission

HPI students learn methods of well-structured development of even very complex software systems, always taking best readability and transparency into consideration.

Software developers, according to the HPI mission, shall be able to communicate there thoughts, understand existing software systems and make them easily understandable.

Hypertext Transfer Protocol

HTTP is a client-server protocol by which two machines can communicate over a tcp/ip connection. An HTTP server is a program that sits listening on a machine's port for HTTP requests. An HTTP client (we will be using the terms HTTP client and web client interchangeably) opens a tcp/ip connection to the server via a socket, transmits a request for a document, then waits for a reply from the server. Once the request-reply sequence is completed, the socket is closed. So the HTTP protocol is a transactional one. The lifetime of a connection corresponds to a single request-reply sequence. (a transaction)

HTTP is the protocol used for document exchange in the World-Wide-Web. Everything that happens on the web, happens over HTTP transactions. TCP/IP networking and HTTP are the two essential components that make the web work. In order to write software that accesses the web (like a web browser, or a custom web client) you need a basic understanding of both.

P

PaSaMuF is a word first used and probably invented by Professor Siegfried Wendt. He used it as an example of something that has a name but no sense. Lars Trieloff, one of our project managers, had the idea that this word is still in search of a meaning and our product was in search of a name. He gave it the meaning Publish and Share and Manage ur Files.

R

Requirement

A requirement is a “function or characteristic of a system that is necessary...the quantifiable and verifiable behaviors that a system must possess and constraints that a system must work within to satisfy an organization#s objectives and solve a set of problems”

Similarly, “requirement” has the following definitions [IEEE 90]:

1. a condition or capability needed by a user to solve a problem or achieve an objective;

2. (2) a condition or capability that must be met or possessed by a system

or system component to satisfy a contract, standard, specification, or other formally imposed documents;

3. (3) a documented representation of a condition or capability as in (1) or (2).

S

Simple Object Access Protocol

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

T

Apache Tomcat

Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

Tomcat is developed in an open and participatory environment and released under the Apache Software License. Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project.

W

Webservices

Webservices are a set of software interfaces to make programs accessible by exchanging XML data via HTTP.

> The same way programmatic interfaces have been available since the early days of the World Wide Web via HTML forms, programs are now accessible by exchanging XML data through an interface, e.g. by using SOAP Version 1.2 [http://www.w3.org/TR/soap12-part1/], the XML-based protocol produced by the XML Protocol Working Group [http://www.w3.org/2002/ws/Activity#xmlp-wg].
>
> — W3C Web Services Architecture Working Group

More information related to Webservices can be found at the W3C Webservices Activity Statement [http://www.w3.org/2002/ws/].

Appendix C. License

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). This product also includes software that is licensed under the terms of the GPL

The Apache Software License, Version 1.1

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

Warning

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/>

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.

GNU General Public License

Version 2, June 1991

Free Software Foundation, Inc.

59 Temple Place, Suite 330,
Boston,
MA 02111-1307
USA

.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and

2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work un-

der copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License.

Exception:

If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not per-

mit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY

COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.