Training plan overview

Secure file storage

Prepared		
Revieved		
Approved		

Document history

Version	Date	Author(s)	Description
0.1			Initial draft

Contents

1. Purpose of the project

The goal of the second phase of the Linux programming training is to design, implement, integrate and test ***secure file storage server and client*** applications.

Each application shall be modular, with well-defined interfaces. The main proposed packages for each application are shown in Image 1.

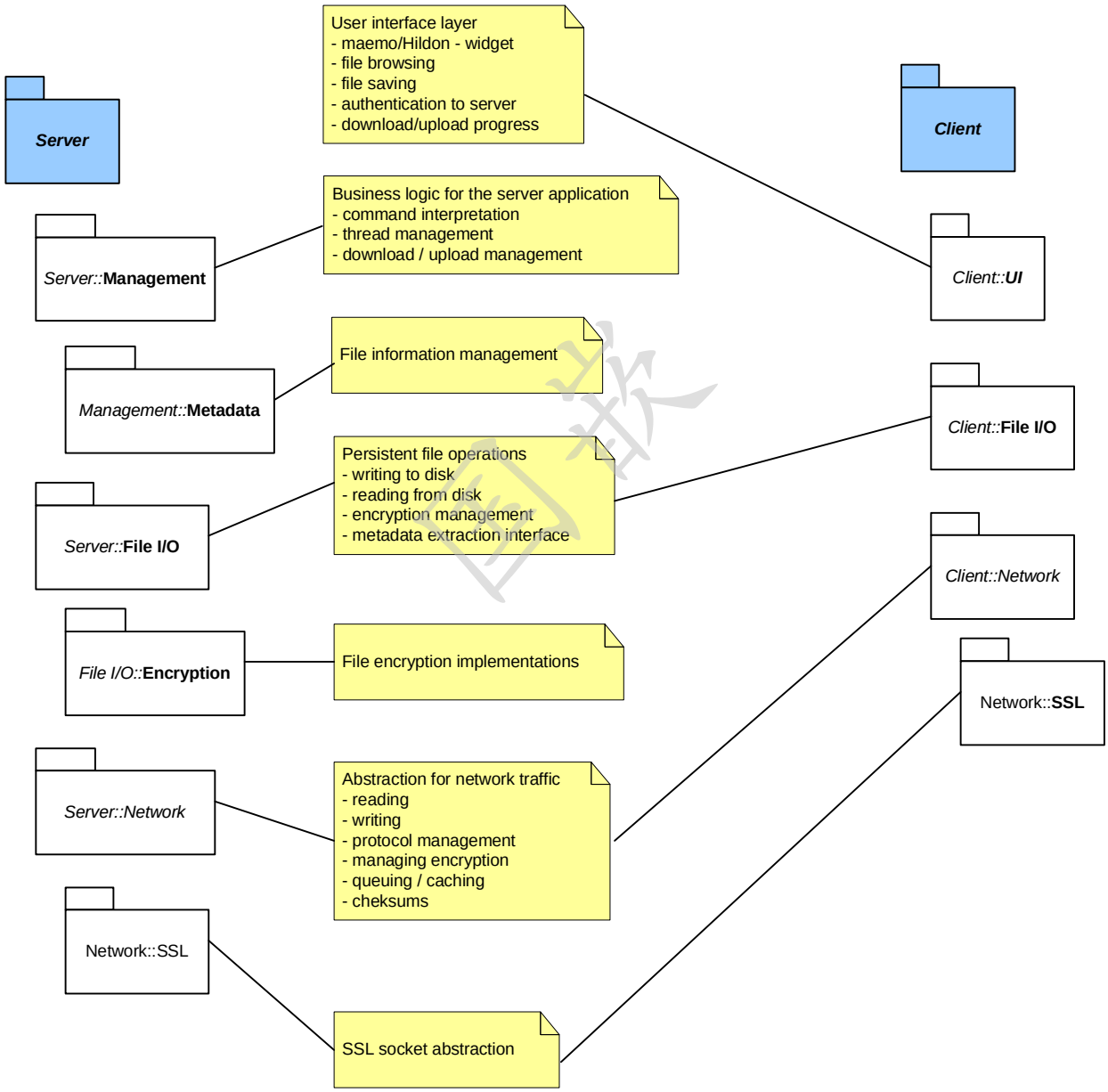


Image 1. Main packages

2. Requirements

2.1 Requirement types

There are three types of requirements scoped out in this document that define what features is required from the Secure File Storage. The following words are used in each requirement to indicate the level of importance:

MUST HAVES: the requirement must be fulfilled, i.e. it is obligatory.

SHOULD HAVES: the requirement should be fulfilled. The system is preferred to fulfil the requirement but it is not obligatory.

MAY HAVES: The requirement may be fulfilled. The requirement indicates a possible feature or functionality as a suggestion.

2.2 Requirement list

2.2.1 General

1) The system MUST enable the storage of a file sent from the client to disk using strong file encryption
2) The system MUST support attaching metadata to each file to be stored
3) The system MUST support retrieval of file metadata from the server
4) The system MUST enable the retrieval of a file stored to the disk to the client
5) The system MUST use secure network transfer
6) The system MUST have defined command/transfer protocol in application level
7) The command/transfer protocol MUST have a file integrity check
8) ...

2.2.2 Server

1) The server MUST be multi-threaded, allowing multiple simultaneous client connections
2) The server SHOULD be configurable to adjust the maximum amount of client connections
3) The server MUST use SSL server authentication
4) The server SHOULD allow changing the SSL certificate over existing established connection
5) The server MUST be configurable using settings file

6) The server SHALL be run-time configurable using setup commands
7) The server MUST log the file transaction events
8) The server MUST support username/password -authentication
9) The server MAY support OS-level authentication from clients
10) The server MAY support re-establishing of downloads/uploads in case of connection failure
11) ...

2.2.3 Client

1) The client MUST be a qualified maemo- application with defined behavior regarding taskbars, termination etc.
2) The client MUST have file browsing capability for both upload and download
3) The client MUST have metadata entering and displaying capability
4) The client MUST ensure successful file download, i.e. check for available disk space etc.
5) The client MUST be able to connect using SSL sockets
6) The client MUST support username/password –authentication to the client
7) The client SHOULD support server management features, such as setting connection limits, changing the certificate etc.
8) ...

3. Personnel

3.1 Roles

Each programming team will have following roles:

- Server Lead
- Client Lead
- Integration Lead
- Programmer
- Server integrator
- Client integrator

3.1.1 Server lead

Server lead is responsible for server design desicions together with the server team. Additionally, server lead shall take part in server programming.

3.1.2 Client lead

Client lead is responsible for client design desicions together with the server team. Additioanlly, client lead shall take part in client programming.

3.1.3 Integration lead

Integration lead is responsible for:

- integration schedule
- integration plan
- verification plan
- version management planning

Additionally, integration lead shall take part either server or client programming.

3.1.4 Programmer

Programmer shall implement the required features, create unit test scripts and perform unit testing prior submitting code to integration.

3.1.5 Server integrator

Server integrator is responsible for ensuring that:

- all server code submitted to version management by the end of the day is compilable
- all designated unit tests are created
- run the unit test scripts before committing the code to daily integration
- design the unit test scripts with the responsible programmer

Additionally, server integrator shall take part on server programming.

3.1.6 Client integrator

Client integrator is responsible for ensuring that:

- all client code submitted to version management by the end of the day is compilable
- all designated unit tests are created

- run the unit test scripts before committing the code to daily integration
- design the unit test scripts with the responsible programmer

Additionally, client integrator shall take part on client programming.

3.2 Programming team

The team should be composed of ~ 10 persons, with following role break-up:

- Server Lead
- Client Lead
- Integration Lead
- Programmer 1
- Programmer 2
- Programmer 3
- Programmer 4
- Programmer 5
- Server integrator
- Client integrator

4. Schedule and management

4.1 General

The project schedule and progress is managed by Plenware staff, and the responsible person will act as project manager for all development teams.

4.2 Management procedure

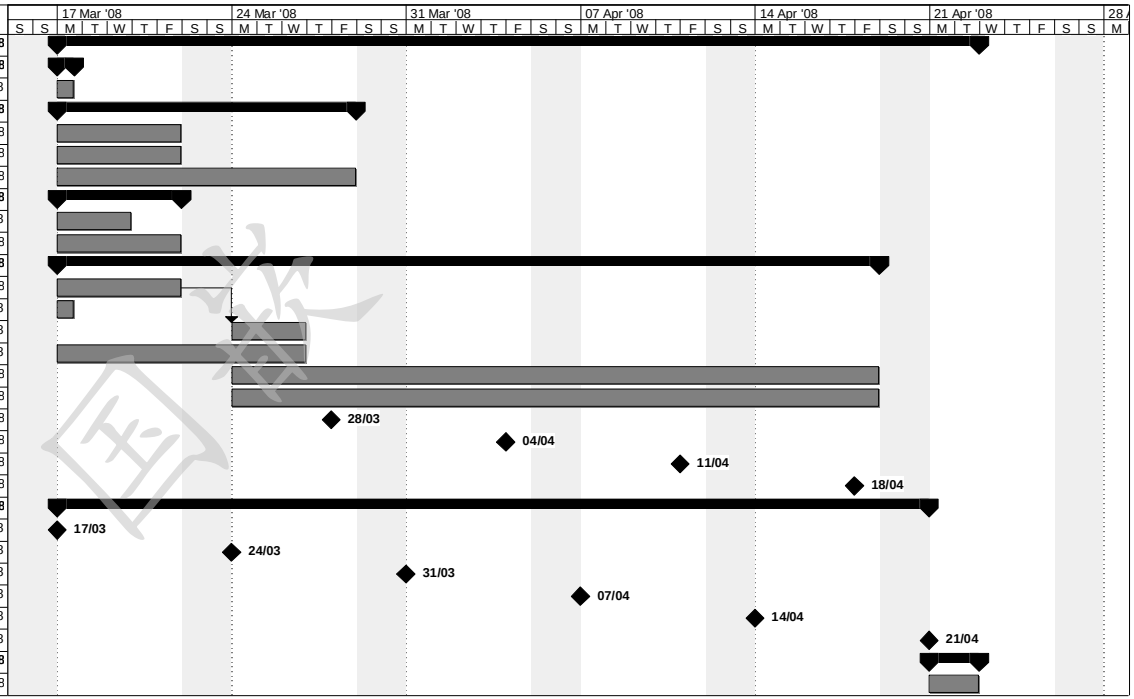
The project will be ran as a 'lightweight' AGILE/SCRUM project, with following key points:

- Project will start with a kick-off, where main features are discussed in casual manner
- All teams will implement all the high priority features
- After initial planning, the programming is done in weekly cycles
    - Feature implementation order may be changed during the process, if necessary
- In the beginning of each programming week there is a team meeting with project manager

  - previous week achievements
  - feature backlog overview
  - progress monitoring
- All code is integrated daily, with unit tests
- Every day, there will be a 15-20 minute team meeting prior daily integration
  - Integration criteria is decided

4.3 Schedule

ID	Task Name	Duration	Start	Finish
1	**Secure file storage**	**28.8 days?**	**Mon 17/03/08**	**Tue 22/04/08**
2	**Establishing the project**	**1 day?**	**Mon 17/03/08**	**Mon 17/03/08**
3	Kick-off	1 day?	Mon 17/03/08	Mon 17/03/08
4	**Orientation**	**10.67 days?**	**Mon 17/03/08**	**Fri 28/03/08**
5	OpenSSL	5.33 days?	Mon 17/03/08	Fri 21/03/08
6	Encryption (Blowfish etc.)	5.33 days?	Mon 17/03/08	Fri 21/03/08
7	maemo	10.67 days?	Mon 17/03/08	Fri 28/03/08
8	**Environment set-up**	**5.33 days?**	**Mon 17/03/08**	**Fri 21/03/08**
9	Version management	3.2 days?	Mon 17/03/08	Wed 19/03/08
10	Scratchbox environment	5.33 days?	Mon 17/03/08	Fri 21/03/08
11	**Implementation**	**26.67 days?**	**Mon 17/03/08**	**Fri 18/04/08**
12	Architecture planning	5.33 days?	Mon 17/03/08	Fri 21/03/08
13	Architecture review	1 day?	Mon 17/03/08	Mon 17/03/08
14	Interface design	3.2 days?	Mon 24/03/08	Wed 26/03/08
15	Integration planning	8.53 days?	Mon 17/03/08	Wed 26/03/08
16	Server programming	21.33 days?	Mon 24/03/08	Fri 18/04/08
17	Client programming	21.33 days?	Mon 24/03/08	Fri 18/04/08
18	Integration 1	0 days	Fri 28/03/08	Fri 28/03/08
19	Integration 2	0 days	Fri 04/04/08	Fri 04/04/08
20	Integration 3	0 days	Fri 11/04/08	Fri 11/04/08
21	Integration 4	0 days	Fri 18/04/08	Fri 18/04/08
22	**Administration**	**26.67 days**	**Mon 17/03/08**	**Mon 21/04/08**
23	Project meeting 1	0 days	Mon 17/03/08	Mon 17/03/08
24	Project meeting 2	0 days	Mon 24/03/08	Mon 24/03/08
25	Project meeting 3	0 days	Mon 31/03/08	Mon 31/03/08
26	Project meeting 4	0 days	Mon 07/04/08	Mon 07/04/08
27	Project meeting 5	0 days	Mon 14/04/08	Mon 14/04/08
28	Project meeting 6	0 days	Mon 21/04/08	Mon 21/04/08
29	**Closing the project**	**2.13 days?**	**Mon 21/04/08**	**Tue 22/04/08**
30	Evaluation	2.13 days?	Mon 21/04/08	Tue 22/04/08



Appendices

- App. 1
- App. 2
- ...