Funky Torrents

Software Integration Plan

Version 1.0
IT University of Göteborg
TIG029 – Software Architecture for Distributed Systems

1. Revision History

Version	Date	Description	Author
0.1	2010-12-02	Integration Plan Created	Ali Issa
0.2	2010-12-03	Introduction written	Ali Issa
0.2	2010-12-07	Roles and Responsibilities written	Ali Issa
0.3	2010-12-08	Integration strategy written	Ali Issa
0.4	2010-12-09	Integration strategy modified	Ali Issa
0.5	2010-12-15	Individual Steps and Test Description written	Ali Issa
0.6	2010-12-16	Environment written	Ali Issa
0.7	2010-12-20	Modified Environment	Ali Issa
0.8	2010-12-21	Revision history added and reviewed all parts	Ali Issa
1.0	2010-12-22	Final review before submission	Ali Issa, Savas Aydin

Table 1. Revision history

Table of Content

2. Introduction

This Funky Torrents Integration Plan is taken from the ProjectConnections template [3] This document will provide a strategy for the integration process, describing the tasks to the team members in each iteration step. The tools, roles and responsibilities concerning the integration process are defined in this documentation. It also defines the entry criteria, the criteria that should be fulfilled before starting the integration process and also when the application is considered to be complete.

3. Integration Strategy

3.1.Entry Criteria

Each module has to pass all tests conducted by the test manager in order to be integrated, no form of integration will occur without the approval of the test manager. To make integration more efficient, Team members will be briefly informed of potential issues that might occur while integrating, by the test manager.

3.2.Elements to be integrated

The modules to consider in the integration process are

- check
- commfunc
- connector
- encoder
- filedata
- pars,pass
- GUI with Chicago boss, which is a framework for Erlang.

3.3.Integration Strategy

The integration will be conducted according to *the vertical integration method*; the modules will be integrated according to their functionalities. This method was selected due to the fact that time is the only constrain applied to the funky torrent project. When using the vertical integration method, only the necessary modules are involved which means integration takes less time to

perform. The disadvantage of this method is that cost of owner-ship could be higher than other integration methods. The purpose of this project is educational, for that reason there are no concerns regarding the disadvantage of this particular integration method.

3.4.Integration Sequence

For more detailed information on the iterations, please see the SPMP [1]. For a complete list of responsibilities please see TheGroup_GanttChart [2].

Iteration 1		
Task	**Description**	**Responsibilities**
Initial version of SPMP		
Initial version of SAD		
Intitial version of SDS		
Bittorrent protocol learning		
Erlang programming language learning		
Software Architecture design learning		

Iteration 2		
Task	**Description**	**Responsibilities**
Update project plan		
Update architecture description		
Parse torrent file		
Handshake protocol		
Connect to web server		
Create a database		
Write failing tests		

Iteration 3		
Task	**Description**	**Responsibilities**
Connect to tracker		
Log in functionality		
Chat functionality		
Search functionality		
Write failing tests		

Iteration 4		
Task	**Description**	**Responsibilities**
Write downloading file into hard drive		
Implement necessary modules to download from shell		
Functional interface		
Merge HTML and Erlang codes		
Download through web interface		
Implement additional functionalities ie. Chat, search, login		
Testing	Testing separate modules	
Integration		
Integration testing	Testing the product of the integration	

Iteration 1

The team members will attend several courses in order to obtain knowledge needed to work with the funky torrents project.

Iteration 2

The team members will practically still be in the learning phase, where different type of knowledge is acquired to be able to proceed with the project.

Iteration 3

The team members will focus on implementation, some minor tests will be conducted, no integration at all.

Iteration 4

In this iteration all modules will be tested separately and then integrated, if the modules pass the tests they will be ready for integration. After integration, the test manager will verify that the application delivers the services as intended by conducting integration testing.

4. Individual Steps and Test Description

4.1.Software Integration Test Description

As mentioned earlier, the test manager will test all software modules separately in order to verify that the behaviors of these modules are as expected. This procedure will minimize the frequency of errors during the integration process. When integration is completed, the product of the integration will be tested to verify that the behavior is as expected.

4.2.Subsystem Integration Test Description

There will be two subsystems to integrate in this specific project. Chicago boss which is a MVC framework for Erlang,in this case considered as a subsystem. It will be integrated with some specific software modules which in turn is a part of the second subsystem. Connection between the subsystems will be conducted via a web server which constitutes the user interface. When

integration is complete, the tester will interact with the user interface to test the behavior of the product.

4.3.Final Functional Tests

The test manager and developers will implement an automated function which will be used to call all necessary functions to run the entire program. By running the automated function, the test manager will determine if the application is capable of performing according to the requirements specified for the project.

5. Environment

5.1. Tools and Test Equipment Required

Software tools required for testing are;

- Erlang – concurrent programming language
- Eunit - testing framework for Erlang

5.2.Program Stubs and test Data Required

Stubs will be implemented in case of need to temporarily substitute a yet-to-be-developed code to simulate a specific behavior, in order to test the modules. Stubs will also be implemented, in case of integration were a module is not yet implemented, but the modules input/output is required f to determine if the integration will succeed at that specific iteration.

5.3.Responsibilities and schedule

5.3.1. Roles and Responsibilities

The following team members will be involved in the integration activities:

- Ali Issa - responsible for planning tests and integration
- Savas Aydin - responsible for planning integration
- Michal Musialik - integration developer
- Aydin Halilov-integration developer
- Björn Eriksson - integration developer

- Magnus Bergqvist- integration developer

5.3.2. Key Dependencies

The software modules which were specified previously (see section 2.2) should be implemented according to the specified software module/modules for each iteration. The software modules must be available during integration, in order to follow the integration schedule. Software tools for testing (see section 4.1) must be available in order to test the functionality and behavior of the modules, integration depends on testing as earlier mentioned, all modules must pass the tests conducted by the test manager in order to be integrated.

5.3.3. Risks and Assumptions

Inexperience of working with the current programming language used (Erlang), combined with a time constraint applied on the project makes the risk of complications to occur during the integration process high. Each team member is assigned a specific role within the group; not fully understanding the responsibilities that follows each role is a potential risk. If the roles are not followed, the development process which include the integration process, will be unorganized and inefficient, on the other hand if assigned roles are followed in a rigidly strict manner, the cooperation within the team which is essential in all phases of the development process, but especially in the integration process would suffer.

5.4.Problem Recording and Resolution

If complications occur during integration, the team members will inform each other of the situation verbally during meetings and work hours or via email. In case of urgent matter, communication will be conducted via telephone.

5.5.Rework, Review and Retest Procedures

If the team feels that a review, rework and retest are needed, the team will inform the project manager which in turn assigns the tasks to appropriate team members depending on their role.

5.6.Suspension, Restart and Exit Criteria

Modules that are not in a testable state will be suspended. The tester and the developers must come to an agreement and decide that the modules are in a testable state, in that case the suspension will cease and tests will be conducted. When integration tests has been conducted successfully, which means that the application provides all required services, the application will exit the test stage and will be considered as a complete prototype.

6. References

- [1] Savas Aydin,"Software project management plan", The group, 2010
- [2] Savas Aydin, "Gantt chart" The Group", 2010.
- [3] Integration plan template: Project Connections, www.Projectconnections.com/knowhow/subsets/sample-templates/integration plan.doc.