



**Applications in Practical High-End Computing - Group Project**

Assignment - "Workflow"

**Test**

Supervisor: Dr Stuart Barnes

Authors: Mateusz Gołąb

Csaba Kerti

Jakub Kiełbasa

Zsolt Kollarits

Course: CSTE / SETC

**Table of contents**

[1 Summary 3](#_Toc318587825)

[2 Requirement test 3](#_Toc318587826)

[3 Preparing Acceptance Test 3](#_Toc318587827)

[3.1 Trace-ability Matrix for use-cases 3](#_Toc318587828)

[3.2 Trace-ability Matrix for test-cases 4](#_Toc318587829)

[4 References 5](#_Toc318587830)

# Summary

All software probably contains errors. To reduce errors and faults we tried to use software driven development, and execute test operation after every steps of the software develop project.

# Requirement test

It is very important to find misunderstands and faults of a software at the early stage of the project as soon as possible. Furthermore many projects fail because the misinterpreted requirements. So it is very important to clarify the requirements with the customer.

The [1] web page gives many good ideas how to check and review the requirements of a software project. There is no standard method to execute this test. It is usually a review made by the stakeholders and if it is any change, the customer should accept it if there is a change.

We checked the user requirements that are they clear and consistent. It should be clear for that designer or programmer who read it at the first time and doesn’t know the aims of the project. We had many assumptions at the beginning of the project, and after day by day we tried to discover them. We contacted with the customer and clarified these assumptions. We read and reviewed the given document many times to clarify missing and hidden requirements.

The non-functional requirements are very important too. Every non-functional requirement should contain concrete values to be measurable when the test will be executed.

# Preparing Acceptance Test

## Trace-ability Matrix for use-cases

The acceptance test should validate the user needs and requirements. It should be executed at the end of the project but it is recommended to start prepare it at the beginning.

After we had produced the requirements we have created the use-cases using the UML. The Trace-ability Matrix is a good practice to check that is all the functional requirements are covered by use-cases.

We have created a Trace-ability Matrix to ensure that all requirements should be covered by uses-cases. The columns contain the requirements while the rows contain the uses-cases. When a uses-case connects with a requirement we sign it with a dot.

The Trace-ability Matrix is very usually to follow the requirements changes. Using it is easy to see which use-case should be changed too.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | FU1 | FU2 | FU3 | FU3.1 | FU3.1 | FU4 | FU5 |
| UC\_U1 Log in |  |  |  |  |  | ● |  |
| UC\_U2 Log out |  |  |  |  |  |  |  |
| UC\_U3 Stop Simulation |  |  |  |  |  |  | ● |
| UC\_S1 Run Simulation |  | ● |  |  |  |  |  |
| UC\_S2 Upload Simulation parameters |  |  |  |  |  |  |  |
| UC\_S3 Download results of a Simulation |  |  |  |  |  |  |  |
| UC\_S4 Check status/log of Simulation |  |  |  | ● |  |  |  |
| UC\_A1: Add new module to Workflow | ● |  |  |  |  |  |  |
| UC\_A2: Remove module from Workflow | ● |  |  |  |  |  |  |
| UC\_A3: Provide input/output metadata (XSD) of a Module |  |  |  |  |  |  |  |
| UC\_A4: Provide XML file with parameters and commands |  |  |  |  |  |  |  |
| UC\_A5: Provides XML file with recovery configuration |  |  |  |  |  |  |  |
| UC\_A6: Create new User |  |  |  |  |  |  |  |
| UC\_A7: Inactivate User |  |  |  |  |  |  |  |
| UC\_M1: Configure recovery mechanism |  |  |  |  | ● |  |  |
| UC\_M2: Create backup |  |  | ● |  |  |  |  |
| UC\_M3: Recover |  |  | ● |  |  |  |  |

1. Table – Trace-ability matrix for use-cases

Where the user requirements are the followings:

* FU1 User can add/remove arbitrary number of modules into workflow.
* FU2 User can run simulation with uploaded parameters.
* FU3 Recovery system: possibility of restarting workflow (from the last stable/good point) when system crushes.
  + FU3.1 Monitoring of errors: Users can see the exact location of failures.
  + FU3.2 Flexible recovery policy (depending on expected time of execution we decide to store data before/after module or after each iteration)
* FU4 Many users have possibility to connect to system simultaneously. But there is only one running program at time (users requests' go to queue - serial workflow).
* FU5 User can stop simulation.

## Trace-ability Matrix for test-cases

The Trace-ability Matrix is very similarly usable to check that is exists a test-case for every test-cases. The columns contain the uses-cases while the rows contain the test-cases. When a test-case connects with a use-case we sign it with a dot.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC\_U1 | UC\_U2 | UC\_U3 | UC\_S1 | UC\_S2 | UC\_S3 | UC\_S4 | UC\_A1 | UC\_A2 | UC\_A3 | UC\_A4 | UC\_A5 | UC\_A6 | UC\_A7 | UC\_M1 | UC\_M2 | UC\_M3 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

2. Table - Trace-ability matrix for test-cases

# References

|  |  |
| --- | --- |
| [1] | <http://www.softwaretestinghelp.com/how-to-test-software-requirements-specification-srs/>, 2012 |
| [2] | Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black : Foundations of software testing |
| [3] | Brian Hambling : "SOFTWARE TESTING - An ISTQB–ISEB Foundation Guide", 2nd Edition, 2010 |
| [4] | Rex Black, Jamie Mitchell: Advanced Software Testing Vol. 3, 2011 |