



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

Assignment - "Workflow"

**Requirements**

Supervisor: Dr Stuart Barnes

Authors: Mateusz Gołąb

Csaba Kerti

Jakub Kiełbasa

Zsolt Kollarits

Course: CSTE / SETC

**Table of contents**

[1. Summary 3](#_Toc318408746)

[a) Communication with Customer 3](#_Toc318408747)

[b) Impact on following stages of project 3](#_Toc318408748)

[2. System's Dictionary 4](#_Toc318408749)

[a) Actors 4](#_Toc318408750)

[b) Layers 4](#_Toc318408751)

[c) Access layer 5](#_Toc318408752)

[d) Control layer 5](#_Toc318408753)

[e) Calculation layer 5](#_Toc318408754)

[3. User Requirements 6](#_Toc318408755)

[a) Functional requirements 6](#_Toc318408756)

[b) Non-functional requirements 6](#_Toc318408757)

[4. System Requirements 7](#_Toc318408758)

[a) System diagram 7](#_Toc318408759)

[b) Manage Simulation diagram 9](#_Toc318408760)

[c) Manage Modules diagram 10](#_Toc318408761)

[d) Manage Users diagram 11](#_Toc318408762)

[e) Recovery Mechanism diagram 12](#_Toc318408763)

[Appendix A: Used Tools 13](#_Toc318408764)

[a) Enterprise Architect ver. 9.2 Trial Ultimate 13](#_Toc318408765)

[b) Google Code 13](#_Toc318408766)

1. Summary

Preparation of complete requirements list is a first and, according to many authorities, most difficult part during project life. Any misunderstanding or even inexactness can cause very serious complications in next parts of the project. Each pound saved on requirements analysis may come back as a dangerous difficulty in future and will cost tens times more.

Different methodologies choose different approaches to this issue. Some, older ones believe in inflexible boundaries of project parts, where after finishing requirements module comes design and amendments to previous part are not allowed. Others, more modern ones assume that changes in requirements may happen. In this group we can distinguish extreme ones which actually assume that changes in project specification are certain, and team members have to accept them and be able to give quick responses. Our team decided to use one of these (Agile) methodologies - Scrum.

1. Communication with Customer

Scrum assumes every-day meetings and very intensive communication with customer applying involving them into project ins and outs. This approach should decrease number of misunderstandings between project team and customer. Unfortunately, because of nature of academic assignments it was impossible to involve customer (lecturer) into every-day Scrums and long-lasting (usually 2-4 hours) discussions after them. Knowing this, we mocked Customer's presence on meetings by writing and receiving e-mails to and from them. This form of dialog, especially at the beginning helped us in defining specific requirements placed in next chapter in different form (functional, non-functional system).

1. Impact on following stages of project

Well-defined requirements are base of creating scalable and reasonable design. Knowing that we have tried to decrease amount of time spent on work in parallel on both of them. The problems were twosome, firstly Scrum itself assumes that requirements will change and secondly very short time period devoted for this assignment (less than 3 weeks) not allowed us to completely exhaust a subject. Second problem was independent from us so we decided to do our best within given time resources. For dealing with first issue we tried to distinguish project skeleton very early and focus on it, so that it would be fixed when design part starts and accept the fact that smaller features may come and go.

Requirements have also direct and indirect impact on test plan. Well-define use cases give possibility of creating reliable acceptance tests and traceability matrix.

Requirements' impact on implementation is only indirect (through design).

1. System's Dictionary

In order to avoid as many misunderstandings as possible, authors decided to create System Dictionary. All, listed below, terms are being used in all following project's stages. Because of that fact, each change in it would cause many, difficult-to-detect complications, so we decided to put, especially lot of effort in creation of as short as possible but unambiguous definitions. For distinguishing them we adopted the notation with terms starting with capital letter.

1. Actors

**User** - person who uses System.

**Scientist** - person who can use Workflow to run simulation through Terminal using Client.

**Administrator** - person responsible for configuration of the Workflow.

**Workflow Manager** - program which controls Workflow. Responsible for sending commands and parameters to particular parts.

**Recovery Manager** - program which performs all tasks in order to assure safe recovery in case of crush during execution modules of Workflow Sequence.

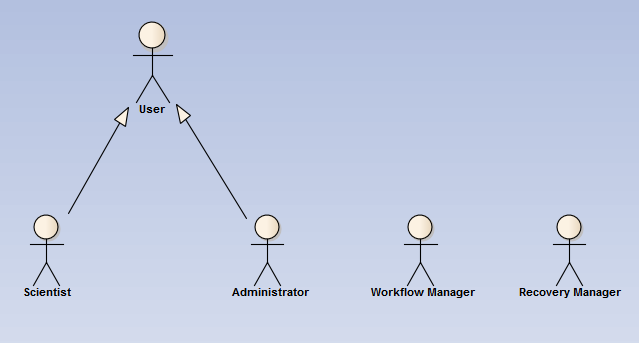


Diagram 1: dependencies between actors.

1. Layers

**System** - all resources and programs used in order to achieve project's aims. Organised, logically in three layers: access, controlling and calculating one.

1. Access layer

**Client** - program which gives Scientist possibility to remotely run simulation or Administrator to change settings of Workflow.

**Terminal** - place where Client runs.

1. Control layer

**Workflow** - all entities: programs and modules controlling and performing simulations scientist task.

**Workflow Server** - logical machine where Workflow runs.

**Workflow Sequence** - ordered, connected list of programs performing actual work of system. Sequence of steps necessary to finish task. Set up by Administrator.

**Simulation** - all start parameters needed in Workflow Sequence plus scientist id. May be considered as a instance of Workflow Sequence.

**Simulations Queue** - queue of Simulations in Workflow sent by Scientist waiting for execution.

**Module** - single unit-program within Workflow Sequence added by Administrator.

1. Calculation layer

**Calculation Server** - logical place where modules' calculations are being performed.

1. User Requirements
2. Functional requirements

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.

* 1. FU3.1 Monitoring of errors: Users can see the exact location of failures.
  2. 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).

1. Non-functional requirements

NF1 Reliability/Validation: take care of input/output formats of specific modules.

NF2 Use universal/independent communication standards between modules

NF3 Data flow should be based on XML files.

NF4 Modules can run on many different remote systems using different platforms but hey have to be already installed.

NF5 Network connection between terminal, workload server and remote systems are established.

1. System Requirements
2. System diagram

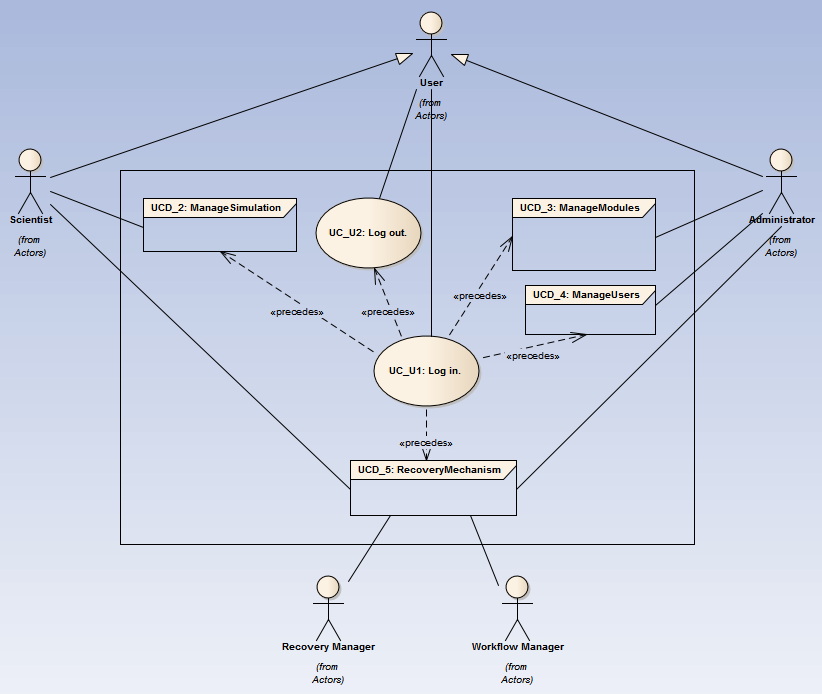


Diagram 2: use case diagram of entire System.

* **UC\_U1: Log in.**

|  |  |
| --- | --- |
| **Description:** | Cziki cziki |
| **Requirements:** | **-** |
| **Constraints:** | **-** |
| **Basic Path:** | "Successful log in."  1. User connects through client to system.  2. User types credentials on log in screen.  3. User gain access to system resources. |

* **UC\_U2: Log out.**

|  |  |
| --- | --- |
| **Description:** | Cziki cziki |
| **Requirements:** | **-** |
| **Constraints:** | **-** |
| **Basic Path:** | "Successful log out."  1. User logs out from system. |

1. Manage Simulation diagram

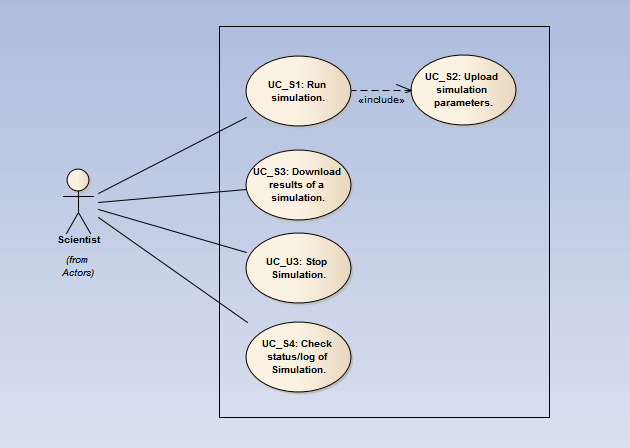


Diagram 3: use case diagram of Simulations' Management.

* **UC\_S1: Run Simulation.**
* **UC\_S2: Upload Simulation parameters.**
* **UC\_S3: Download results of a Simulation.**
* **UC\_U3**: **Stop Simulation.**

|  |  |
| --- | --- |
| **Description:** | Cziki cziki |
| **Requirements:** | **-** |
| **Constraints:** | **-** |
| **Basic Path:** | "Successful log out."  1. User logs out from system. |

* **UC\_S4: Check status/log of Simulation.**

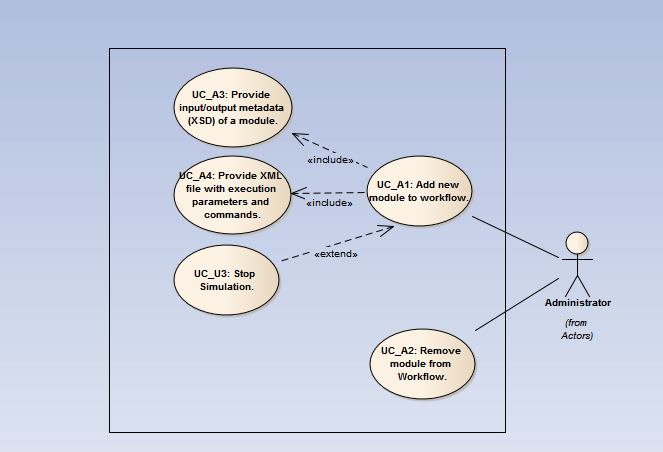
1. Manage Modules diagram

Diagram 4: use case diagram of Modules' Management.

* **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 execution parameters and commands.**

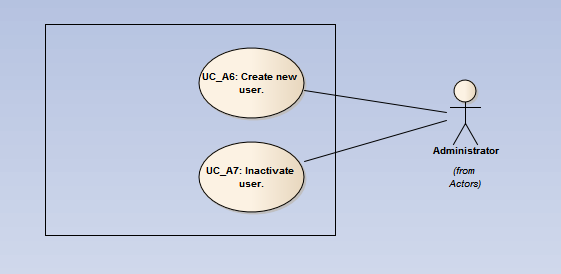
1. Manage Users diagram

Diagram 5: use case diagram of Users' Management.

* **UC\_A6: Create new User.**
* **UC\_A7: Inactivate User.**

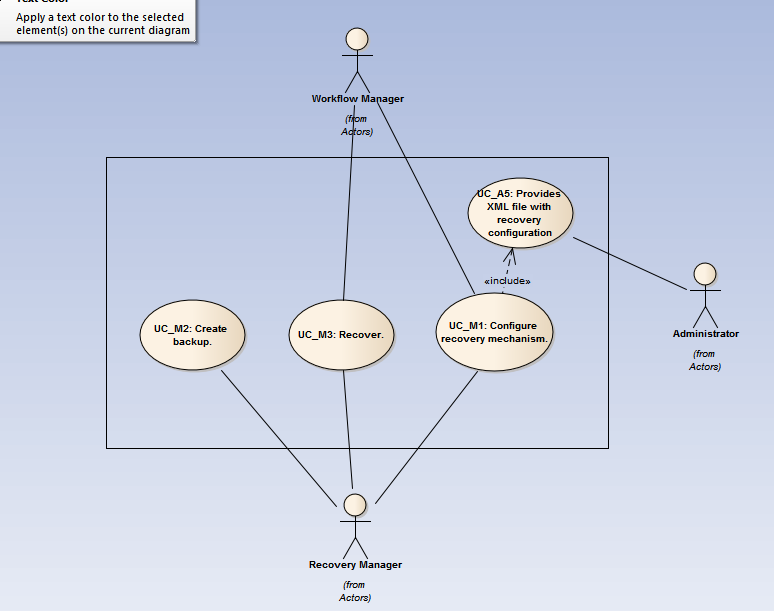
1. Recovery Mechanism diagram

Diagram 6: use case diagram of Recovery Mechanism.

* **UC\_A5: Provides XML file with recovery configuration**
* **UC\_M1: Configure recovery mechanism.**
* **UC\_M2: Create backup.**
* **UC\_M3: Recover.**

Appendix A: Used Tools

1. Enterprise Architect ver. 9.2 Trial Ultimate
2. Google Code