## cranfield.jpgCRANFIELD_logo.gif

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

Assignment - "Workflow"

**Design Document**

Supervisor: Dr Stuart Barnes

Authors: Mateusz Gołąb

Csaba Kerti

Jakub Kiełbasa

Zsolt Kollarits

Course: CSTE / SETC

## Table of contents

[Introduction 3](#_Toc318139736)

[Stucture of the system 3](#_Toc318139737)

[1. Overview of the structure 3](#_Toc318139738)

[2. Detailed explanation of the components 4](#_Toc318139739)

[Modules 4](#_Toc318139740)

[Workflow manager 4](#_Toc318139741)

[Recovery manager 4](#_Toc318139742)

[Workflow queue 4](#_Toc318139742)

3. [The list of concrete configuration files 4](#_Toc318139742)

[Design patterns 6](#_Toc318139743)

[Control flow 6](#_Toc318139744)

[Data 6](#_Toc318139745)

[Resource 7](#_Toc318139745)

[Presentation 7](#_Toc318139745)

## Introduction

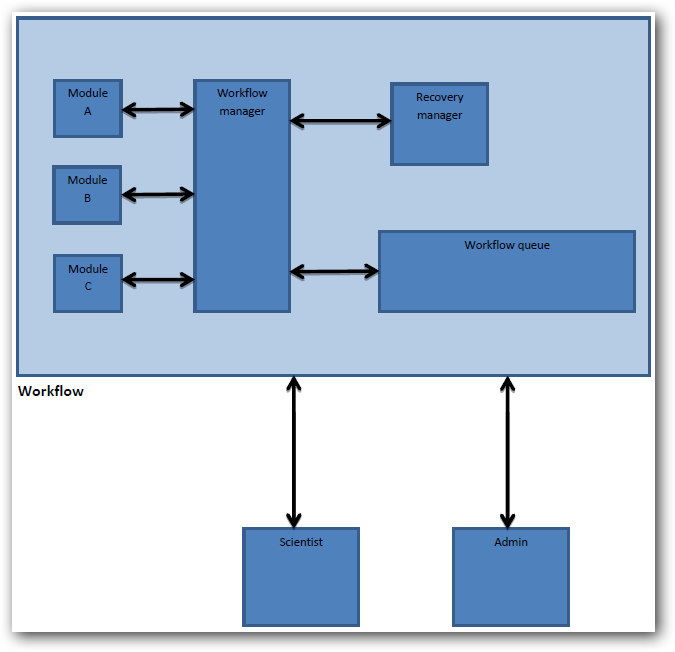
This document contains the detailed design data of the application. It shows the components of the program and the patterns which implement them. Information can also be found about the Recovery System.

## Structure of the system

1. **Overview of the structure:**

The program consists of a couple of individual components, each has a differenttask. *These parts are:*

1. **Modules**
2. **Workflow manager**
3. **Recovery manager**
4. **Workflow queue**
   1. Format validation files (XSD)
   2. Module metadata files (commands/parameters XML)



This figure shows the components of the system and the types of users.

1. **Detailed explanation of the components:**

* *Modules:* executable programs/scripts,
* defined by Administrators (they are able to replace the modules and change the parameters - corresponding xml file)
* run by Scientists (they run the pre-set modules - workflow – and use the results)
* *Workflow manager:* Basically the program’s core is the Workflow manager which manages the running of modules and keeps the connection with the Recovery manager. The program is responsible for run the specific modules and monitor/validate the proper execution. TODO
* *Recovery manager:* The Recovery manager implements one of the most important feature of the program – correction of the system when error occurred.

Possible errors:

* *module failure:* Problems with the modules (execution problems).
* *timeout:* It occurs when the response time of a module or resource is too long.
* *resource unavailability*: If a necessary resource is (temporarily) unavailable.

Possible reactions from the system:

* *rollback:* Process should be undone to the last stable point in workflow.
* *suspend:* Process should stop when error occurred N times in the same point of workflow.
* *Workflow queue:* This part of the system contains the necessary information for module running (i.e. xml files with parameters, validation files).

*Format validation files:* XSD file is located between each module which checks the output of the pre-module and ensures appropriate input for the post-module.

*Module metadata files:* Contains the module data required for running. (Module start-up parameters.)

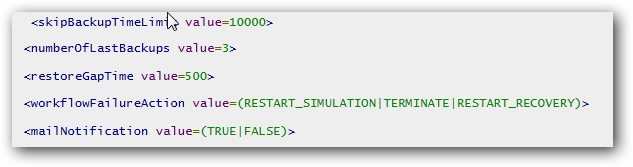
1. **The list of concrete configuration files:**

(where N=number of modules in the workflow and M=number of simulations in the Workflow queue)

conf.xml: Contains parameters for the main application.

Number of files: *1*

Contents:



where,

*skipBackupTimeLimit*: if a module is short enough the program does not create a backup.

*numberOfLastBackups*:

*restoreGapTime*:

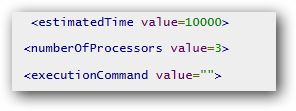
*workflowFailureAction*:

*mailNotification*:

moduleParameters.xml: Contains bunch of information for each module.

Number of files: *N*

Contents:



where,

*estimatedTime:*

*numberOfProcessors:*

*executionCommand:*

validator.xsd: XSD file for validate input and output of each module.

Number of files: *N*

scientistParameters.xml: XSD file for validate input and output of each module.

Number of files: *M*

initialParametersSkeleton.xml: Structure of input parameters of first module in the workflow sequence.

Number of files: *1*

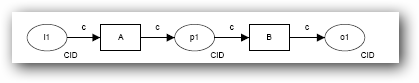
## Design Patterns

The design of the system based on general design patterns. There are 4 individual part of designing which required appropriate design patterns:

* *Control flow:*

The program implements the **Sequential workflow pattern**, it was the most reasonable choice for this program. A task in a process in enabled after the completion of a preceding task in the same process. The workflow controls the sequence of activities and decides which of the steps will execute next. Consider sequential workflows if you must execute a series of predefined steps to accomplish a certain task. The sequence pattern is used to model consecutive steps in a workflow process and is directly supported by each of the workflow management systems available. The typical implementation involves linking two activities with an unconditional control flow arrow.

The next figure shows the structure of this pattern, where A and B are the *modules* and l1, p1, o1 are *the format validation parts*.

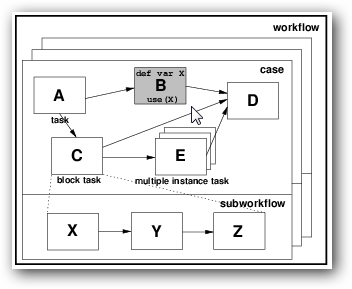


For animation: <http://www.workflowpatterns.com/patterns/control/basic/wcp1_animation.php>

* *Data:*

The most appropriate pattern in this case the **Task Data pattern,** whose main feature is that data elements can be defined by tasks which are accessible only within the context of individual execution instances of that task.

Let’s see a figure about this pattern, where A,B,C,D,E are tasks:



For animation: <http://www.workflowpatterns.com/patterns/data/visibility/wdp1_animation.php>

* *Resource:*

The base of the program’s resource handling is **the Automatic execution pattern.** The most important property is the ability for an instance of a task to execute without needing to utilise the services of a resource. Where a task is nominated as automatic, it is initiated immediately when enabled. Similarly, upon its completion, subsequent tasks are triggered immediately.

For animation: <http://www.workflowpatterns.com/patterns/resource/creation/wrp11_animation.php>

* *Presentation:*

TODO

