## 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

[1](#_Toc318390341)

[Table of contents 2](#_Toc318390342)

[1. Introduction 3](#_Toc318390343)

[2. Architecture 3](#_Toc318390344)

[2.1. Workflow components 3](#_Toc318390345)

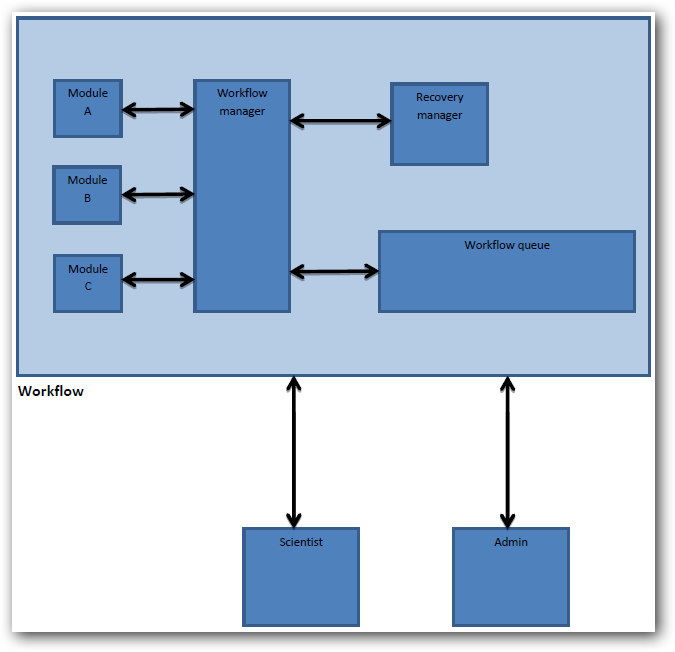
[Design Patterns 5](#_Toc318390346)

# Introduction

This document contains detailed design information of Workflow system. Includes design patterns, architectural models, Workflow configuration and recovery mechanism description.

# Architecture

System is designed according to client – server architecture. Server part comprises of Workflow and there are two types of clients , Scientist client and Administrator client which both have access to Workflow via terminal.



Workflow outline :

## Workflow components

* **Module** - component which represents executable program/script, which is running in a workflow. Administrator provides every module to the Workflow together with :
* Module metadata files (commands/parameters XML)
* Format validation files (XSD)
* **Workflow manager** *–* Main controller which manages all Workflow components. Responsible for running simulations as well as validating and monitoring each module. Decides when to use Recovery Manager.
* **Recovery manager -** The Recovery manager implements one of the most important feature of the Workflow – recovery mechanism. This component manages backup and restore mechanisms and keeps logic of Workflow recovery policy.
* **Workflow queue** *-* This component logically represents queue containing necessary parameters for module execution , provided by Scientist.

# Design Patterns

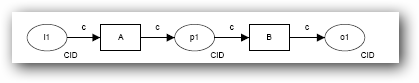
When designing workflow application we can consider group of special design patterns specified for such type of programs. Such group comprises of control flow patterns, data patterns and resource patterns. On workflow component level we can consider using universal design patterns for software development.

## 3.1 Workflow specific design patterns

**Control flow pattern : Sequential workflow pattern**

The idea of this pattern is to enable a task in a process 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. Such pattern is appropriate for developing 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.

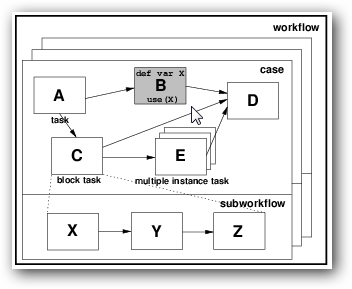
The next figure shows the structure of this pattern, where A and B are the *modules* and l1, p1, o1 are *the format validation parts*.



**Data pattern : Task Data**

Main idea of this pattern is that data elements can be defined by tasks. Every task is accessible only within the context of his individual execution instance. This pattern suits perfectly requirements of workflow we are developing. According to requirements every module in workflow has his own data to operate on. In our case there is no need to create and manage shared data space for every module placed in workflow . Every executed module operate on his own data.

Gray color represents data ,which is accessible only to B module.



**Resource pattern : Automatic execution**

The most important property of this pattern is the ability for an instance of a task to execute without needing to utilise the services of a resource. The mechanism of the program’s resource handling can be implemented using this pattern**.** Where a task is nominated as automatic, it is initiated immediately when enabled. In other words, program executed as a module within workflow manages all available resources provided by system.

## 3.2 Application specific design patterns

# Recovery mechanism

## 4.1.Backup policy

Performing a backup of stable point in Workflow Sequence is essential activity to enable recovery mechanism. Decision when backup should be done is made dynamically. Execution time of every module is measured and compared with value of configurable parameter. If module’s execution time is greater than parameter’s value, backup of Workflow Sequence state will be performed. It is possible that every module’s execution time will be less than specified in parameter. Therefore after every iteration, Workflow manager checks how many backups were made in iteration. If none, Recovery manager will perform backup. Such mechanism ensures at least one backup per iteration and allows dynamic backup policy.

## 4.2. Recovery

The main aim of recovery mechanism of the Workflow system is to react on failure caused by module, by restarting workflow from some previous stable point in Workflow Sequence. When error occurs in Workflow Sequence , Workflow Manager starts recovery mechanism by launching Recovery Manager . First activity of this component is receiving last stable point of Workflow Sequence from Database Manger and sending it to Workflow Manager to restart Workflow Sequence. When error occurs after these activities , Recovery Manager waits recovery gap time which is specified by Administrator in configuration file and then again provides last stable point to Workflow Manager. This gap between recoveries gives possibility to wait if there is some problem with remote server which hosts a module, instead of performing another recovery instantly. This mechanism is configurable , so Administrator can decide whether it is useful for actual workflow. When failure occurs again, previous stable point is provided after recovery gap. Scenario of using previous stable point in case of error is repeated. Total number of recoveries is limited by parameter in configuration file. If workflow failed despite recovery attempts one of the actions specified by parameter is undertaken. Possible actions are : restarting whole simulation, terminating simulation or performing recovery process again.

# Configuration

Administrator has possibility of configuring some Workflow and Recovery mechanisms using configuration files.

## 5.1 conf.xml

Contains Recovery parameters.

Content:



This parameter describes time in seconds which is compared to execution time of every module . If execution time is less than parameter, backup will be skipped .



Represents number of stable points recovery attempts within one recovery process.



Describes time in seconds which elapses between every single recovery attempt.



This parameter represents action which is undertaken after unsuccessful recovery process.

* RESTART\_SIMULATION - simulation is restarted.
* TERMINATE – simulation is terminated, next simulation from the Worker queue starts.
* RESTART\_RECOVERY – recovery process is restarted.

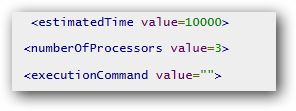


This parameter controlls email notification mechanism. Notification can be successfully finished simulation or workflowFailureAction value.

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*

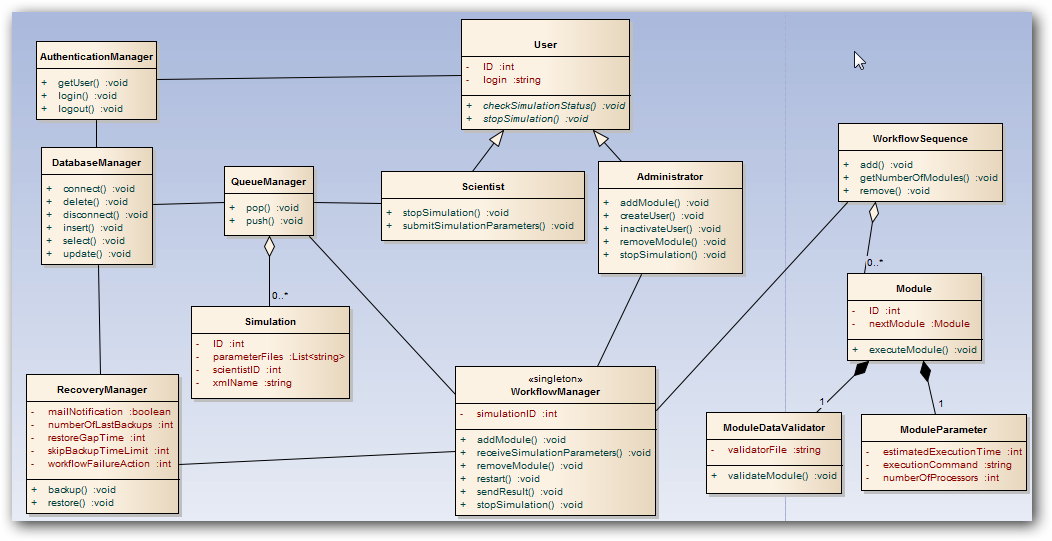
scientistParameters.xml: XSD file for validate input and output of each module.

Number of files: *M*

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

Number of files: *1*

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



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.