**Software Engineering Process**

**Software Requirements Document**

Version 0.1 (draft), September 12, 2012

**{Put LOGO instead of text}**

|  |  |  |
| --- | --- | --- |
| **Project Team:** | Marco Antonio Ramirez Balbuena | 14452 |
|  | Beatriz Medina Rodriguez | 17049 |
|  | Christian A. Thierry Garcia | 17055 |
|  | Gilberto M. Langarica Quintero | 16927 |
|  | Cintya G. Landa Lagunas | 16958 |
|  | Rodolfo A. Contreras Ramirez | 17079 |
|  | Miguel Alejandro Velez del Callejo | 17031 |
| **Project Manager:** | Arlin Grijalba Gomez | 17330 |
| **Advisor:** | Dr. Adán Hirales Carbajal | CETYS University |
| **Co-advisor:** | LCC Cristal Dena Flores | MTIC-UABC |
|  |  |  |
| **Customer:** | Dr. Adán Hirales Carbajal | CETYS University |

ICC, CETYS University

Abstract

This document contains the software requirements for the Deterministic Rostering Scheduler (DRS) that enables personnel and resource allocation. The document complies with the Software Requirements Document (SRD) from the Software Engineering Standard RUP.

**Table of contents**

[Introduction 5](#_Toc336267442)

[1.1 Purpose 5](#_Toc336267443)

[1.2 Scope 5](#_Toc336267444)

[1.3 List of definitions and abbreviations 5](#_Toc336267445)

[1.3.1 Definitions 5](#_Toc336267446)

[1.3.2 Abbreviations 5](#_Toc336267447)

[1.4 Documents 6](#_Toc336267448)

[1.4.1 Reference Documents 6](#_Toc336267449)

[1.5 Overview 6](#_Toc336267450)

[General Description 7](#_Toc336267451)

[2.1 Background 7](#_Toc336267452)

[2.2 Environment 7](#_Toc336267453)

[2.2.1 System requirements 7](#_Toc336267454)

[2.3 Problem definition 7](#_Toc336267455)

[2.3.1 The scheduling heuristic 8](#_Toc336267456)

[2.3.2 The use case 12](#_Toc336267457)

[1.2.1 Glossary 12](#_Toc336267458)

[1.2.2 Acronyms and Abbreviations 13](#_Toc336267459)

[2 General Description 13](#_Toc336267460)

[2.1 Context of the Class Scheduler 13](#_Toc336267461)

[2.3 General constraints on Class Scheduler 16](#_Toc336267462)

[2.4 General assumptions and dependencies 16](#_Toc336267463)

[2.5 User characteristics 16](#_Toc336267464)

[3.1 Constraints 16](#_Toc336267465)

[3.3 Functional Requirements 19](#_Toc336267466)

Document Status Sheet

|  |  |
| --- | --- |
| Document title | Software Requirements Document |
| Document Identification | TGSF/Documents/SRD/0.1 |
| Author(s) | <your names> |
| Version | 1.0 |
| Document Status | draft / internally accepted / conditionally approved / approved |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary** |
| 0.1 | 09-09-2012 | Rodolfo A. Contreras Ramirez  Christian A. Thierry Garcia | Document creation |
| 0.1.1 | 9-24-2012 | Adán Hirales Carbajal | Mayor revision |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Chapter 1

# Introduction

## 1.1 Purpose

The purpose of this document is to identify the architectural model of the Teikoku Grid Scheduling Framework (tGSF) and to present user requirements for school time tabling support.

## 1.2 Scope

The software will be used to implement a time tabling scheduler. Currently tGSF is able to allocate jobs to a set of parallel machines, also referred to as resources. With time tabling support, tGSF will allocate a set of jobs (subjects) to a set of resources (rooms), while considering employee availability and qualifications restrictions.

## 1.3 List of definitions and abbreviations

### 1.3.1 Definitions

|  |  |
| --- | --- |
| Term | Description |
| TODO |  |
|  |  |
|  |  |

### 1.3.2 Abbreviations

|  |  |
| --- | --- |
| Term | Description |
| DTTS | Deterministic Time Tabling Strategy |
| RRS | Resource Reservation Strategy |
| TcR | Tasks with Reservations |
| TcF | Tasks with Failures |
|  |  |
| tGSF | Teikoku Grid Scheduling Framework |

## 1.4 Documents

### 1.4.1 Reference Documents

|  |  |
| --- | --- |
| [ESA] | ESA Software Engineering Standards (ESA PSS-05-0 Issue 2), ESA Board for Software Standardization and Control (BSSC), 1991 |
| [LKA] | Leung, J.; Kelly, L. & Anderson, J. H. Handbook of Scheduling: Algorithms, Models, and Performance Analysis CRC Press, Inc., 2004 |
| [SML] | Smith-Miles, K. & Lopes, L. Measuring instance difficulty for combinatorial optimization problems Computers &amp; Operations Research, 2012, 39, 875 - 889 |
| [HAS] | Hassan Gomma, Designing Concurrent, Distributed, and Real-Time Applications with UML, Addison Wesley, 2000 |
| [HTY] | Hirales-Carbajal, A.; Tchernykh, A.; Yahyapour, R.; González-García, J. L.; Röblitz, T. & Ramírez-Alcaraz, J. M. Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid Journal of Grid Computing, Springer -Verlag, Netherlands, 2012 |
| [HTR] | Hirales-Carbajal, A.; Tchernykh, A.; Roblitz, T. & Yahyapour, R. A Grid simulation framework to study advance scheduling strategies for complex workflow applications Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, 2010, 1 -8 |

## 1.5 Overview

This document is structured according to the standard described in [ESA]. In chapter two a description of the system and its environment is given, along with a model that described the software and charter three contains the software requirements.

Chapter 2

# General Description

Proponemos una heurística para el problema de determinístico de calendarización de personal a recursos. Restringimos el modelo del problema tal que los recursos no son compartidos por múltiples grupos colegiados, por ende, son administrados únicamente por un único grupo colegiado. Se conoce que problema de calendarización de personal a recursos es NP-completo. A menudo métodos combinatorios son empleados para su solución pero incurren en alto costo computacional, por lo que muchos estudios prestan atención en mejorar el desempeño del algoritmo de calendarización. En este trabajo, prestamos mayor atención en derivar criterios que permitan evaluar la calidad de las soluciones generadas

## 2.1 Background

## {TODO: Research}2.2 Environment

The system will run on a Java Virtual Machine (JVM) Ver. 1.6.

### 2.2.1 System requirements

The software application will be further described in section XX, but have the following minimal requirements:

General requirements:

* Windows XP, Mac OS X or Linux 2.4 (or higher)
* SUN JRE 1.42 or 15, as long as the previous operating systems support JVM 1.6

Hardware requirements:

* Intel…

### 2.2.2 System constraints

Since tGSF is based on Java, therefore it requires the Java Virtual Machine, specifically version 1.6.

### 2.2.3 Other constraints

In order to understand this document, the reader must hold knowledge in:

* Software engineering: requirements, see [HAS] chapter 7
* Software engineering: COMET software development process see [HAS] chapters 6.
* Deterministic scheduling problems, see [LKA] chapter 1 and 3.
* The architecture of he Teikoku Grid Scheduling Framework, see [HTY] and [HTR].

### 2.2.4 User characteristics

The primary user is the Academic Coordinator, who is responsible of providing all input data to tGSF. The data includes knowledge of: employees availability; number, time and sizes of available resources.

## 2.3 Problem definition

El problema general de calendarización de personal es formulado de la siguiente manera.

Existe un horizonte de planificación dividido en periodos para . La duración de un periodo es constante, en el dominio del problema un periodo corresponde a una hora. Existen recursos cada recurso tiene su propio horizonte de planificación e inicialmente su horizonte de disponibilidad . No se consideran la capacidad de los recursos.

Dentro del horizonte de planeación, deben ser realizadas tareas . Una tarea es descrita por la tubla con una duración de *horas clases*; *horas laboratorio*, *horas taller*, *horas clínicas*; *tiempo de liberación* e identificador de tarea .

El número de empleados que realizan la tarea en un tiempo es uno. Dado un conjunto de empleados, con , cada empleado está calificado para realizar un subconjunto de tareas. Por ejemplo, implica que el empleado esta calificado para realizar únicamente las tareas 1 y 2. Un *patrón de disponibilidad* del empleado es definido como:

* Un vector de ceros y unos , donde si esta disponible en el tiempo , de otra manera sería .
* Una asignación de tareas de para cada periodo de tiempo ocurre cuando

Un periodo está compuesto por un conjunto de días, de lunes a viernes. Cada día está delimitado por un conjunto de horas laborales, normalmente de 8:00AM a 10:00PM.

Un patrón de trabajo de una ventana es representada como un vector binario , donde si el periodo de tiempo es un periodo de trabajo en el cual la tarea es realizada.

No todos los patrones de trabajo son factibles, ya que un patrón probablemente no satisface una o más restricciones. Consideramos las siguientes restricciones.

1. No existen preferencia de empleados.
2. Los recursos no se comparten entre dos o más programas. Solo se considera la asignación de las tareas (materias) de un programa educativo.
3. Dada la tarea , la cantidad de horas consecutivas impartidas es finita. Tal intervalo de tiempo es denominado Horas Consecutivas Clase (). No se puede permite reservar dos o más en un mismo día.
4. No se permite colocar dos o más en diferentes recursos en un mismo día.

La suma de las duraciones de no debe ser menor o igual a 6.

### 2.3.1 The scheduling heuristic

El simulador tGSF (Teikoku Grid Scheduling Framework) se utiliza para estudiar problemas de calendarización en línea, determinísticos y no determinísticos. Contiene dos niveles de calendarización. En el primer nivel, aplica una estrategia de asignación de tareas la cual selecciona un recurso y hacer llegar la tarea al mismo. Las tareas que arriban a un recurso son colocadas en su cola de espera. En el segundo nivel, sistemáticamente se extraen las tareas de la cola de espera y se aplica una estrategia de calendarización local, por ejemplo: FCFS (First Come First Served) y Easy-Backfilling.

Los requerimientos de una tarea son satisfechos cuando su duración es asignada a uno o más recursos. Durante el proceso de calendarización, la duración de la tarea es fragmentada en bloques de HCC horas. Para cada bloque, un recurso es seleccionado siempre y cuando satisfaga las restricciones 1-4. El anterior proceso de calendarización es denominado RRS (Resource Reservation Strategy).

RRS hace uso de un mecanismo de reservación administrado por un sistema de información. Dado recursos, el sistema de información consulta la disponibilidad de ejecutar la tarea en un intervalo de tiempo dado en cada recurso; selecciona un recurso empleando un criterio de selección, por ejemplo: el recurso con menor número de reservaciones; un recurso aleatoriamente; el recurso con menor o mayor capacidad, etc.; y hace permanente la reservación en el recurso seleccionado. Se asume que el anterior proceso es atómico, por lo que no se permite la interrupción del proceso de reservación. La cancelación de una o más reservaciones existentes solo ocurre cuando no es posible satisfacer una o más restricciones de una tarea. El proceso de reservación se realiza aplicando los siguientes criterios.

1. RRS consulta el identificador del empleado vinculado a y hace el tiempo de liberación de el tiempo de disponibilidad mínimo del empleado .
2. Dada la tarea , RRS itera mientras las duraciones o alguna restricción de no sea satisfecha.
3. RRS selecciona la duración del vector de duraciones de . El vector se recorre secuencialmente.
4. Si la duración es mayor que cero. RRS consulta si la tarea fue asignada previamente a un recurso. Una tarea calendarizada es etiquetada con el identificador del recurso, tal etiqueta es referenciada como .
5. RSS crea un área de búsqueda
6. Si no pertenece al horizonte del empleado , , RSS hace , regresa a 5.
7. Si si pertenece al horizonte de , RRS consulta si existe disponibilidad en el intervalo de tiempo en cada recurso . Sea y sea un vector de disponibilidades no nulas . Note que .
8. Si , es decir existe uno o más recursos con disponibilidad en el intervalo
   1. Si es no nula y , RRS selecciona el recurso
   2. Si es nula o , RRS selecciona un recurso aplicando un criterio de selección. Por ejemplo, el recurso con menor número de reservaciones; aleatoriamente; el recurso con menor o mayor capacidad, etc.
9. RRS hace permanente la reservación en ; reduce la duración de a ; eliminar la disponibilidad del docente en el intervalo ; y modifica el tiempo de arribo de a el siguiente tiempo de disponibilidad del empleado ; hace , (marca para marca como no disponible El tiempo de liberación debe corresponder al siguiente día en el patrón de disponibilidad del empleado.
10. Si es cero, RRS remueve la etiqueta de , regresa a 2.

En caso que los requerimientos de no puedan ser satisfechos, RRS cancela las reservaciones hechas para , de lo contrario el producto de RRS es un conjunto de reservaciones. El seudocódigo 1, ilustra el procedimiento para la creación de reservaciones para una tarea.

La calendarización de una tarea requiere de dos fases: *reservación*, donde dada una tarea RRS intenta crear un conjunto de reservaciones. En la fase de *confirmación*, la tarea es enviada a sus respectivos recursos, haciendo efectiva la reservación.

Dada tareas la *Estrategia Determinística de generación de* Horarios (*Deteministic Time Tabling Strategy, DTTS*) sistemáticamente toma una tarea , aplica RRS y produce un conjunto de reservaciones o ninguna. DTTS mantiene dos bitácoras: *Tareas con Reservaciones* (TcR) y *Tareas con* *Fallos (TcF)*. Las reservaciones generadas por RRS con almacenadas en TcR, mientras que las tareas con fallos en TcF. Una vez todas las tareas fueron procesadas por RRS, DTTS envía las tareas en TcR a sus respectivos recursos, el calendarizador local en cada recurso recibe la tarea y hace efectiva su reservación.

DTTS recibe una bitácora con los trabajos a calendarizar. Cada línea de la bitácora contiene los siguientes campos: horas clase , horas laboratorio , horas taller , horas clínicas , tiempo de liberación , identificador de tarea y el identificador del empleado que está calificado para realizar . DTTS produce un horario en cada recurso. Cada horario contiene los siguientes campos: tiempo de liberación , el tiempo de finalización , identificador de tarea , identificador del empleado , promedio de cancelaciones, utilización por día y el tamaño promedio de horas libres.

**Seudocódigo 1.** RRS( )

**Entrada:** La tarea

**Salida:** Un conjunto de reservaciones o el identificador la tarea que no cumplió con alguna

restricción

1. hacer el tiempo de disponibilidad mínimo de , donde es el empleado vinculado
2. **while** las duraciones o alguna restricción de de no se satisfecha **do**
3. selecciona la duración . Selecciona una duración

mientras no sea cero.

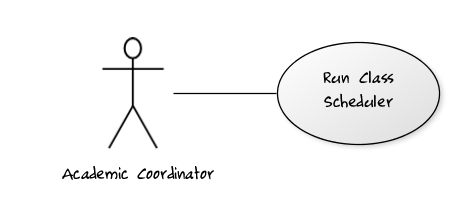
1. **if**  es nula **then**
2. interrumpe la iteración
3. **else**
4. obtén la etiqueta de
5. **endif**
6. crea el área de búsqueda
7. **if** **then**
9. **else**
10. consulta si existe disponibilidad en el intervalo de tiempo en cada recurso ,

almacena en si es no nulo

1. **endif**
2. **if** **then**
3. **if** es no nula y **then**
5. **endif**
6. **if** es nula y **then**
7. selecciona un recurso aplicando un criterio de selección de recursos
8. **endif**
9. **if** es nulo **then**
10. interrumpe la iteración
11. **else**
12. hacer la reservación permanente en el recurso
14. etiqueta el intervalo A en como no disponible
15. el siguiente tiempo disponible del siguiente en
16. **endif**
17. **if** **then**
18. remueve la etiqueta de
19. **endif**
20. **end while**
21. **if** alguna restricción de no es satisfecha **then**
22. deshacer las reservaciones hechas para
23. **endif**

Chapter 3

# The use case



**UC01 Run the Class Scheduler**

**Summary:** The Academic Coordinator will start the application

**Actor:** Academic Coordinator.

**Preconditions:**

* The Coordinator already compiled the data describing the teacher’s availability, tasks to accomplish and available resources in the swf format.
* The Coordinator placed the file in the specified folder in order for **tGSF** to find it and work with it.

**Description:**

1. The user runs the Class Scheduler either by command line or user interface.
2. The Class Scheduler will interpret the swf input file.
3. If the format is correct, it starts working on the scheduling. Else, Alternative A.
4. When Class Scheduler has finished the Scheduling process, it will output a file containing a set of possible schedules. If it is not possible to schedule the tasks, Alternative B.

**Alternatives:**

**Alternative A:**

1. While reading the file, the Class Scheduler finds incomplete information or format errors.
2. The system will prompt the user with an error.

**Alternative B:**

1. It is not possible to schedule the tasks and resources.
2. Class Scheduler will output an additional file containing the elements that the program wasn’t able to allocate.

**Postconditions:**

* One or two files have been created, one with possible schedules and the second (result of Alternative B) will contain the tasks that were not allocated.