Dynamis Inc.

Class Scheduler Requirements

Document Version: 1.0
Document Date: 9/9/2012
Document Status: Draft

Abstract
In the following document is specified the criteria for the scheduling logic and solution evaluation for the deterministic problem of personnel and resource allocation. The problem was restricted to a specific school group, which means that the resources are not shared with any other school group.

Authors: Vélez, M., Contreras, R.

Table 1 Document Change Record

Title:	Dynamis Inc. Class Scheduler Requirements		
Version	**Issue**	**Date**	**Comment**
1	1	9/9/2012	First version of the document describing requirements for the Class Scheduler.

1 Introduction

The Class Scheduler will be an algorithm customization of the Teikoku Framework oriented to solving the problem of class scheduling and classroom allocation for a school group. It's also responsible of evaluating the generated solutions and presenting them in a particular format.

1.1 Purpose of the document

This document presents the Requirements for the Class Scheduler. They shall be the basis for the design and implementation of the Class Scheduler under the context of the Teikoku framework.

The Class Scheduler is a program responsible of generating schedules for school teachers. It must allocate classes in a particular time span and a classroom based on the teachers' specialties and working schedules.

1.2 Glossary, acronyms and abbreviations

1.2.1 Glossary

Availability Pattern
Vector of 0's and 1's $W_e(t)_{t=0}^{T-1}$ where $W_e(t)=1$ if e is available in time $[t, t+1]$, otherwise it will be $W_e(t)=0$.

Gantt chart
Type of bar chart that illustrates a project schedule.

Java
Programming language which derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities.

Period
Set of days, from Monday to Friday.

Schedule
A series of things to be done or of events to occur at or during a particular time or period.

Task
A tuple *(ps_j, pl_j, pc_j, pt_j, r_j, id_j)* with a duration of $2 \leq ps_j \leq 4$ class hours; $pl_j \geq 0$ lab hours; $pt_j \geq 0$ workshop hours; $pc_j \geq 0$ clinic hours; liberation time $r_j = 0$ and a task identifier id_j.

Work Pattern
Binary vector $\pi = (\pi(j,t))$, where $\pi(j,t)=1$ if in the time period $[t, t+1]$ is the period where the **Task** *j* is done.

Teikoku Grid Scheduling Framework
Generic Java based system for the development and application of resource management strategies in computational grids.

1.2.2 Acronyms and Abbreviations

CHC Consecutive Hours Class
DRS Deterministic Rostering Strategy
FCFS First Come First Served
GIS Global Information System
LIS Local Information System
tGSF Teikoku Grid Scheduling Framework

2 General Description

2.1 Context of the Class Scheduler

As mentioned before, Class Scheduler is an adaptation for the Teikoku Framework. This means that the system will not be build out of nowhere, it will be an implementation and customization of the existing scheduler Teikoku.
The adaptation consists mainly in implementing the scheduling algorithm for the problem and the standardization of the input data for the framework, as well as the configuration of the system to solve the problem at hand.

2.1.1 Problem Definition

There will be available resources (classrooms) with their own Scheduling Horizon *[0,T]*, which is divided in constant periods of time *[t,t+1]* for *t=0,1,...,T-1*. These periods of time are constant of 1 hour each.
There are *m* available resources (classrooms), each with an initial Scheduling Horizon of *[0,T]*. *n* tasks must be accomplished within the Scheduling Horizon of the resources *j=1, …, n*. These tasks consist of a tuple *(ps_j, pl_j, pc_j, pt_j, r_j, id_j)* with a duration of $2 \leq ps_j \leq 4$ class hours; $pl_j \geq 0$ lab hours; pt_j workshop hours; pc_j clinic hours; liberation time r_j and a task identifier id_j .
The number of employees $D_j(t)$ who can work on task *j* in a period *[t,t+1]* is one. Given a set of employees *E*, each employee $e \in E$ is qualified to work on a subset Q_e of tasks. Each employee possesses an **Availability Pattern**. A task Q_e for each period *[t,t+1]* ocurrs when in the employee's **Availability Pattern** $W_e(t) = 1$.

2.1.2 Task Scheduling Constraints

Work Patterns will have the following constraints:

1. No employee has preference over others.
2. Resources aren't shared between school groups, only a school program's tasks can be allocated.
3. The amount of CCH cannot be 2 or greater in the same resource.
4. The amount of CCH cannot be 2 or greater in different resources.

2.1.3 Scheduling Heuristic

A task's requirements are satisfied when its duration $(ps_j, pl_j, pc_j, pt_j, r_j, id_j)$ is assigned to 1 or more resources.

During scheduling, the task's duration will be divided into CHH blocks. A resource is selected for each fragment as long as it satisfies the 4 **Work Pattern** constraints (2.1.2). This scheduling process will be referred to as DRS.

DRS will apply a reserve/cancel mechanism administered by a GIS, which must be atomic and will allow it to determine the initial time for a task for each resource. Given *m* available resources, DRS will select the resource with the lowest initial time and will cancel *m - 1* reservations.

Steps to make a reservation:

1. Given **Task** *J*, DRS asks GIS for an estimation of the initial times in the resources.
2. SIG canalizes the request to each LIC. Each resource has a LIC.
3. LIC checks CCH for *J* and makes a reservation in the local schedule of the resource.
4. LIC checks for the initial and ending time of *J* in the local schedule.
5. LIC sends the initial time to GIS.
6. GIS gathers all the initial times in a vector or tuples, each consisting of the initial time and the resource ID.
7. SIG provides the vector of tuples and sends it to DRS.
8. DRS requests for the cancellation of *m - 1* reservations, excluding the tuple with the lowest initial time.
9. DRS tags the **Task** with the ID of the resource and stores it in persistence. The relational database model is shown in Appendix A.

After the reservation stage is complete, the confirmation stage will commence. I this stage, **Tasks** are canalized to their respective resource and they're made effective in said resource. The DRS algorithm is described in Appendix B.

2.2 General capabilities of Class Scheduler

The system will only be able to read input files that are in the swf format and it doesn't provide any tool to generate it, it is only responsible of reading it, interpreting it and working based on it.

2.3 General constraints on Class Scheduler

Since tGSF is written in the Java Programming Language, it is independent of the operating system.The only real limitation of the system is that it requires a Java Virtual Machine installed, which must be of version 1.5 or superior.

2.4 General assumptions and dependencies

There is an implicit dependency to the tGSF since it's going to make use of it to solve the Class Scheduling problem.
It is assumed that the tGSF supports connections to databases.

2.5 User characteristics

Academic Coordinators
They will provide the input data to the Class Scheduler, based on the teachers' disponibility, the available classrooms and the periods of time in which the classrooms are available.

3 Specific Constraints, Use Cases and Requirements

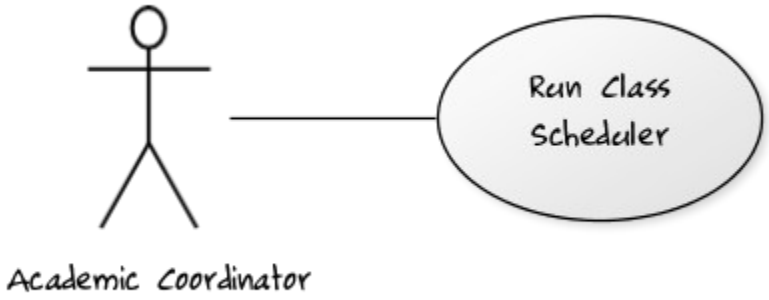
3.1 Constraints

In order to fully understand this document, the reader must have knowledge of the following notations.

CO01 Use Case Diagrams
The reader must be able to understand the basic symbols of UML for Use Case Diagrams.

CO02 Entity Diagrams
The reader must also be familiar with the UML notation for object modeling, in the particular context of database entity diagrams.

3.2 Use Cases



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:

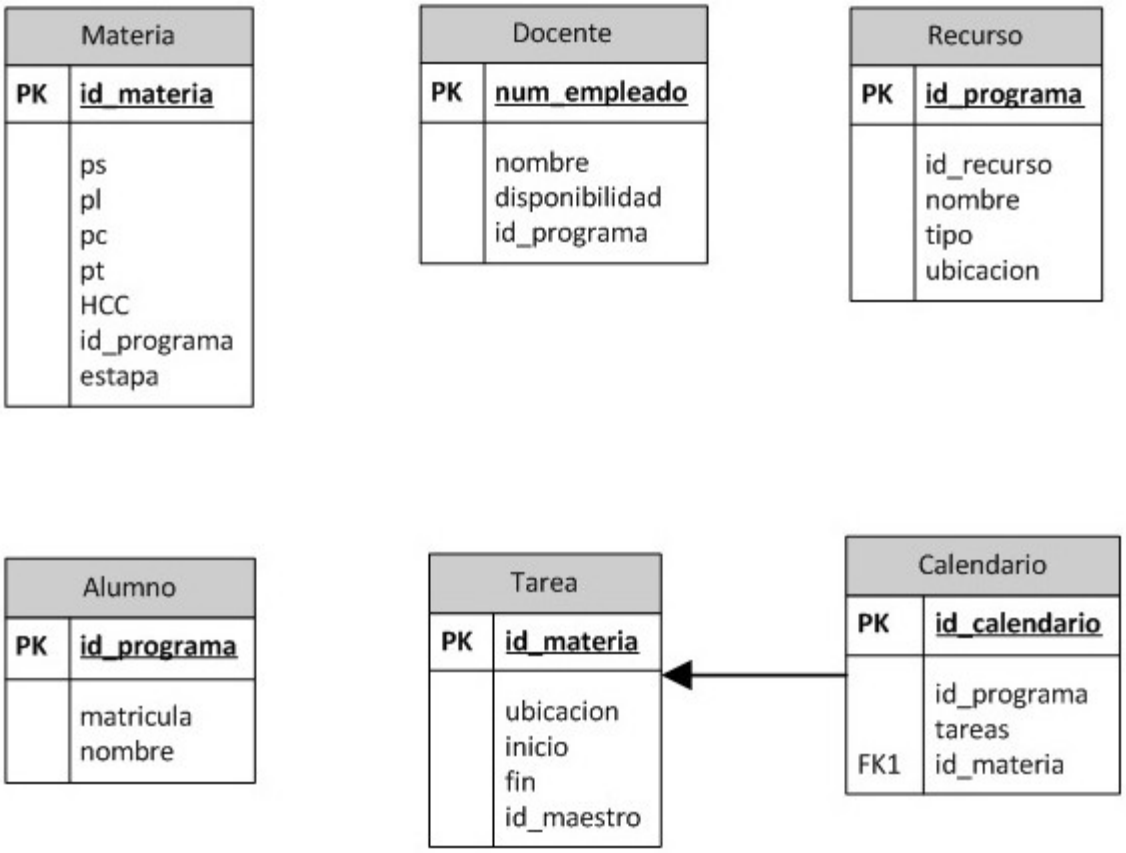
- One or two files have been created, one with possible schedules and the second (result of Alternative B) will contain the tasks that were not allocated.

3.3 Functional Requirements

UR1 DRS Algorithm
The system must implement DRS algorithm.

UR2 Output generation
The system must generate a file with a set of evaluated solutions from which the user may select and implement one.

Appendix A



Appendix B

Input:

- Task set $j = \{ps_j, pl_j, pc_j, pt_j, r_j, id_j\}$
- Employee set E
- Work Pattern of $e \in E$. $W_e(t)_{t=0}^{T-1}$

Start

//Event I: There are no scheduled tasks on initialization

Initialize Hours;

do

//DeadLine estimation phase (earliest termination time)

Estimate earliest deadline and store in DL;

foreach task $n \in V(j)$ **do**

if $t(j).t(p) \geq 0$

//Send component to it's destiny

//Classes classroom

if $t(j).ps_j \geq 0$

TakeClassRoom()

//Block reservation phase

if can **Reserve()**

$R \rightarrow [r_j, r_j + block]$

else

$r \leftarrow r_j + block$

break;

//Employee disposition phase

if $e \in E$ has **Disponibility(*E*, V(e))**

$p_j \leftarrow p_j$ - block

$r_j \leftarrow r_j$ + step

$ps_j \leftarrow ps_j$ - block

//Task assignation to employee phase

AssignTask(Q_e, *E*)

else

//Choose next minimum available hour for the employee

Teacher_Disp($w_e(t)$)

$r_j \leftarrow r_j$ + disponibility

//Lab classes

if $t(j).ps_j \geq 0$

TakeLabRoom()

//Block reservation phase

if can **Reserve()**

$R \rightarrow [r_j, r_j + block]$

else

$r \leftarrow r_j + block$

break;

//Employee disposition phase

if $e \in E$ has **Disponibility(*E*, V(e))**

$p_j \leftarrow p_j$ - block

$r_j \leftarrow r_j$ + step

$pl_j \leftarrow pl_j$ - block

//Task assignation to employee phase

AssignTask(Q_e, *E*)

else

//Choose next minimum available hour for the employee

Teacher_Disp(w_e(t))

$r_j \leftarrow r_j$ + disponibility

//Clinic classes

if $t(j).pc_j \geq 0$

TakeClinicRoom()

//Block reservation phase

if can **Reserve()**

$R \rightarrow [r_j, r_j + block]$

else

$r \leftarrow r_j + block$

break;

//Employee disposition phase

if $e \in E$ has **Disponibility(*E*, V(e))**

$p_j \leftarrow p_j$ - block

$r_j \leftarrow r_j$ + step

$pc_j \leftarrow pc_j$ - block

//Task assignation to employee phase

AssignTask(Q_e, *E*)

else

//Choose next minimum available hour for the employee

Teacher_Disp(w_e(t))

$r_j \leftarrow r_j$ + disponibility

//Workshop classes

if $t(j).pt_j \geq 0$

TakeWorkShopRoom()

//Block reservation phase

if can **Reserve()**

$R \rightarrow [r_j, r_j + block]$

else

$r \leftarrow r_j + block$

break;

//Employee disposition phase

if $e \in E$ has **Disponibility(*E*, V(e))**

$p_j \leftarrow p_j$ - block

$r_j \leftarrow r_j$ + step

$pt_j \leftarrow pt_j$ - block

//Task assignation to employee phase

AssignTask(Q_e, *E*)

else

//Choose next minimum available hour for the employee

Teacher_Disp(w_e(t))

$r_j \leftarrow r_j$ + disponibility

while $|V(j)| \neq 0$

END

Appendix C

Materia	**Type**	**Description**
id_materia	<string + number>	String it's a sequence of two characters indicating the program, the number sequence could be the id or the subject record.
Horas laboratorio	<numeric>	Integer
Horas clase	<numeric>	Integer
Horas taller	<numeric>	Integer
Horas clinica	<numeric>	Integer
Duracion de la sesion	<numeric>	Integer
Programa	<string>	Two or more characters sequence.
Etapa	<string>	Basic, disciplinary and terminal.

Docente	**Type**	**Description**
No. Empleado	<string>	
Nombre	<string>	
Disponibilidad	Vector <int, int>	One int represent the initial time, and the other represent the end time of the Teacher's disponibility. The disponibility begins with the hour zero, which it's the first hour available in the week, and ends in the last hour off the week.
Programa	<string>	Two or more characters sequence.

Recurso	**Type**	**Description**
Id_recurso	<string>	
Nombre_recurso	<string>	
Tipo	<string>	Classroom or lab
Programa	<string>	Sequence of 2 or more characters
Ubicación	<string>	Description where the resources are located, represented by geographic coordinates

Alumno	**Type**	**Description**
matricula	<numeric>	
Nombre	<string>	
Programa	<string>	Sequence of 2 or more characters

Calendario	**Type**	**Description**
Sitio		Represents the place where the subject will be imparted.
Tiempo de inicio		Hour when the subject begins.
Tiempo de finalizacion		Hour when the subject ends.
Id_materia		Subject identificator.

Programa	**Type**	**Description**
Id_programa	<string>	
Nombre	<string>	
adscripcion	<string>	The ascription is the reference to where the program belongs (faculty, etc.)