Class Scheduler Project
Software Architecture Document
Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

Revision History

Date	Version	Description	Authors
16/sep/2012	1.0	<details>	Vélez Miguel, Contreras Rodolfo.

Software Architecture Document

1. Introduction

*[The introduction of the **Software Architecture Document** should provide an overview of the entire **Software Architecture Document**. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the**Software Architecture Document**.]*

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

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.
This document will be concentrated in the following diagrams: Use-Case, Static Modeling, Object Structuring and Statechart modeling.

1.2 Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

The document is influenced by the requirements investigated for the Class Scheduler. It will show diagrams for the development of the next step of the project: design.

1.3 Definitions, Acronyms and Abbreviations

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Architecture Document**. This information may be provided by reference to the project 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 .

Teikoku Grid Scheduling Framework

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

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.

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

1.4 References

*[This subsection should provide a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

Class Scheduler Requirements, 1.0, 9/9/2012, Dynamis Inc.

1.5 Overview

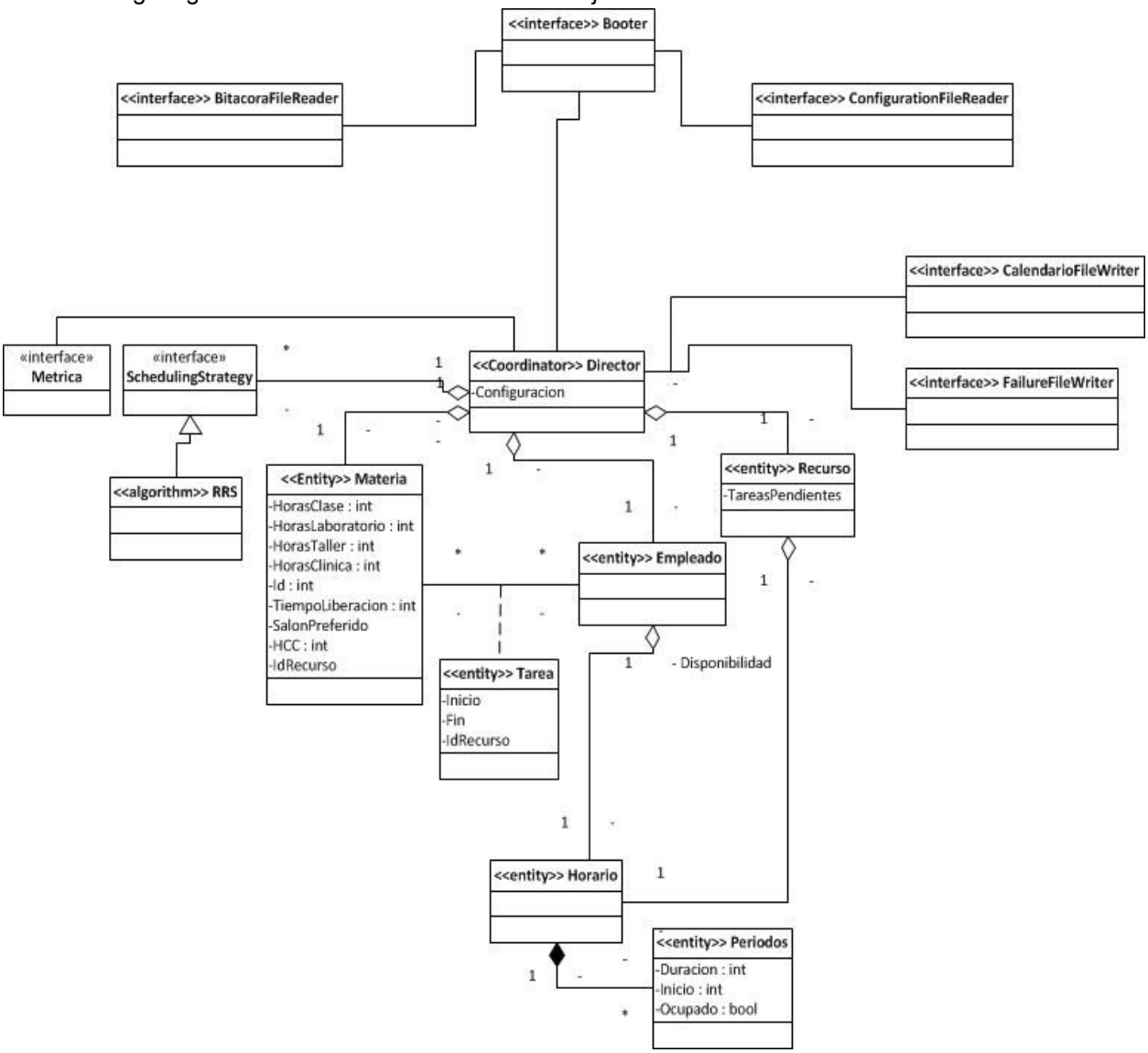
*[This subsection should describe what the rest of the **Software Architecture Document** contains and explain how the **Software Architecture Document** is organized.]*

The document focus specifically in the Collaboration, Class, and Statechart diagrams based on the analysis done with the requirements obtained. It represents to see the main function and many events in the Class Scheduler application. The diagrams were made using the Microsoft Visio program. These will be helpful for the designers in the next step of the project.

2. Architectural Representation

*[This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]*

The following diagram shows the main architectural objects that Class Scheduler consists of.



3. Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

4. Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage - they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.]

4.1 Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

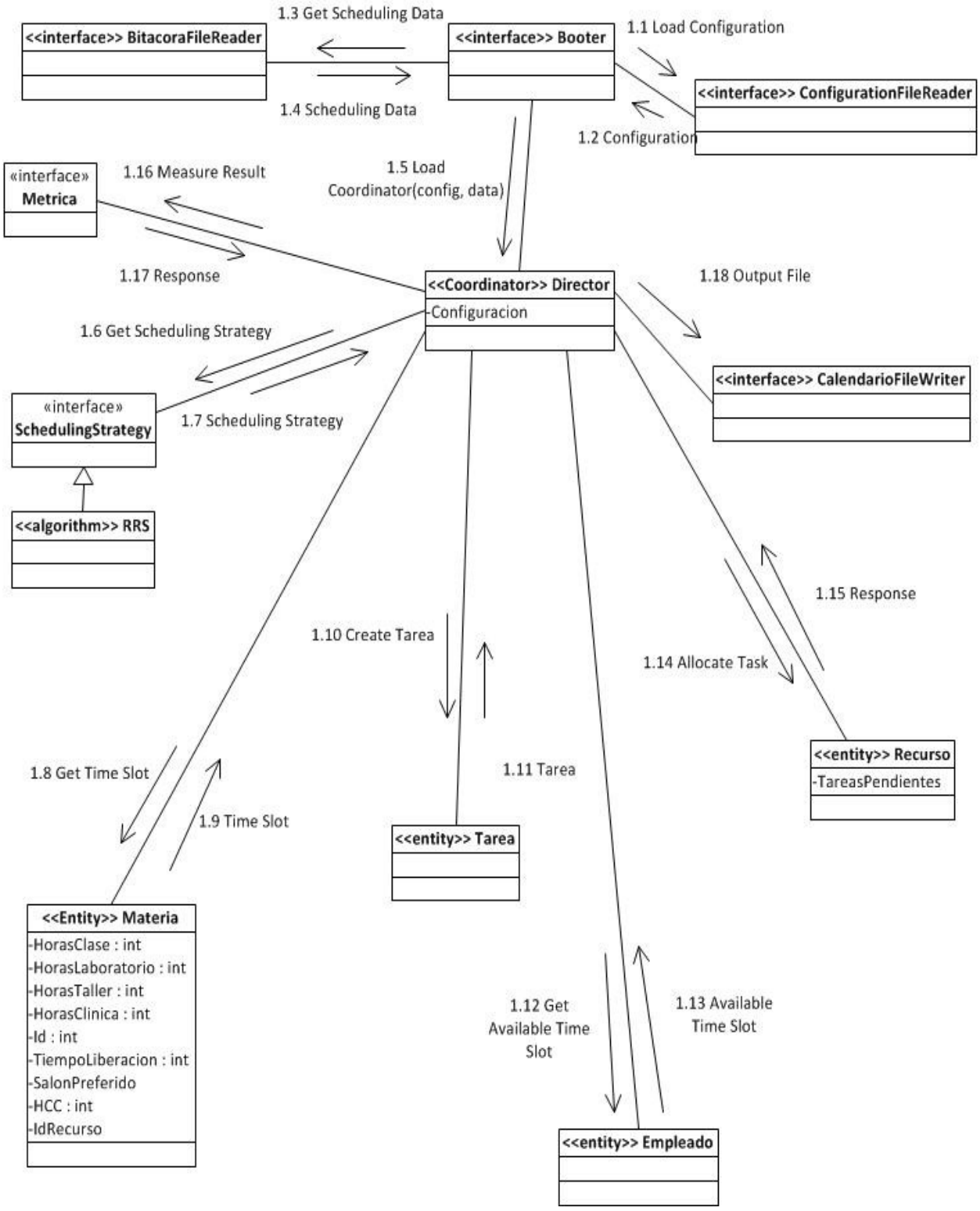
5. Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

In the following collaboration diagram it is showed how the system's classes behave in the basic flow of activity of use case **UC01-Run the Class Scheduler.**

Basic Flow:

1.1 Load Configuration: Load the configuration from the config file and the required elements for said configuration.
1.2 Configuration: The ConfigurationFileReader returns the configuration settings.
1.3 Get Scheduling Data: Load the information from the SWF file.
1.4 BitacoraFileReader returns the workflow data represented as Abstract Data Types.
1.5 Load Coordinator: The Director class is started and configured with the previously obtained data.
1.6 Get Scheduling Strategy: The algorithm that will determine the scheduling logic is loaded based on the configuration.
1.7 Schedulign Strategy: The algorithm is selected.
1.8 Get Time Slot: **[Loop]** Get the next Time Slot that requires allocation.
1.9 Materia returns the next Time Slot, if any.
1.10 Create Tarea: Build the next Tarea with the data obtained.
1.11 Tarea: Returned the created Tarea.
1.12 Get Available Time Slot: Ask for the current Empleado's next available time disposition.
1.13 Available Time Slot: Return Empleado's next available time disposition.
1.14 Allocate Task: Allocate the Tarea in a resource.
1.15 Response: Receive the status of the allocation, it may or may not be able to allocate said Tarea. Go back to step 1.8 until the schedule is complete.
1.16 Measure Result: Use the Metrica object to evaluate the quality of the generated schedule (after allocating all the Tarea objects for all the Materia objects.
1.17 Response: Obtain the resulting evaluation of the generated schedule.
1.18 Output File: Order CalendarioFileWriter to write to disc a file containing the generated schedule.


1.3 Get Scheduling Data
<<interface>> BitacoraFileReader
<<interface>> Booter
1.1 Load Configuration
<<interface>> ConfigurationFileReader
1.4 Scheduling Data
1.2 Configuration
«interface»
Metrica
1.16 Measure Result
1.5 Load
Coordinator(config, data)
1.17 Response
<<Coordinator>> Director
-Configuracion
1.18 Output File
1.6 Get Scheduling Strategy
<<interface>> CalendarioFileWriter
«interface»
SchedulingStrategy
1.7 Scheduling Strategy
<<algorithm>> RRS
1.15 Response
1.10 Create Tarea
1.14 Allocate Task
<<entity>> Recurso
-TareasPendientes
1.8 Get Time Slot
1.11 Tarea
1.9 Time Slot
<<entity>> Tarea
<<Entity>> Materia
-HorasClase : int
-HorasLaboratorio : int
-HorasTaller : int
-HorasClinica : int
-Id : int
-TiempoLiberacion : int
-SalonPreferido
-HCC : int
-IdRecurso
1.12 Get
Available Time
Slot
1.13 Available
Time Slot
<<entity>> Empleado

5.1 Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

5.2 Architecturally Significant Design Packages

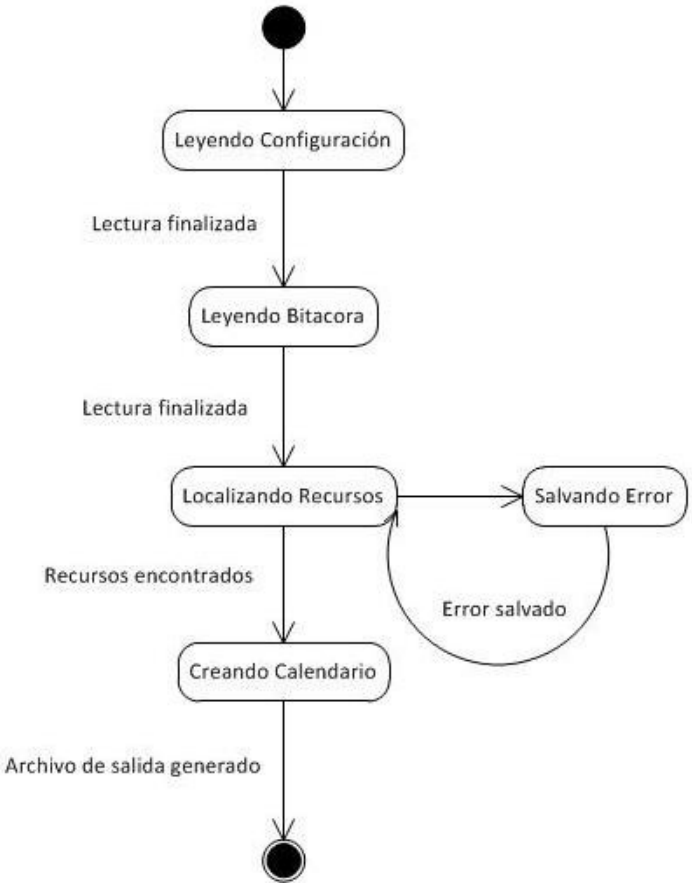
[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.
For each significant class in the package, include its name, brief description, and, optionally a description of some of its major responsibilities, operations and attributes.]

6. Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

The following diagram shows a series of events and processes that will be in the Class Scheduler program.

Leyendo configuración	The program is reading the Coordinator’s configuration.
Leyendo Bitácora	The program finished reading the configuration and is now reading the SWF file.
Localizando Recursos	The program finished reading the SWF file and is now allocating tasks in the resources.
Salvando Error	If the program couldn’t allocate a task, it will save the error and return to Localizando Recursos.
Creando Calendario	The program finished allocating the resources and is now creating the schedule output file.



7. Deployment View

*[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software, and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]*

8. Implementation View

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

8.1 Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers.]

8.2 Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

9. Data View (optional)

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

10. Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

11. Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, for example safety, security or privacy implications, they should be clearly delineated.]