



DOBLE LICENCIATURA EN INGENIERÍA DE  
TELECOMUNICACIÓN Y ADMINISTRACIÓN Y  
DIRECCIÓN DE EMPRESAS

Curso Académico 2014/2015

Proyecto Fin de Carrera

APPLICATION LIFECYCLE MANAGEMENT IN  
MIGRATION PROJECTS

Autor : Cristina Peligros Zarza

Tutor : Dr. Gregorio Robles

Co-tutor : Jonathan Streit



*Dedicado a  
mi familia*



# Agradecimientos

Si cada día agradeceríamos a quienes forman parte de nuestra vida todo lo que tenemos que agradecer, nos ahorraríamos largos ratos de frustración tratando de buscar las palabras y las formas para hacerlo en los acontecimientos importantes.

No es el caso, así que aquí va una lista a todas luces incompleta de agradecimientos a quienes han formado parte de este pequeño logro:

itestra GmbH, por un entorno envidiable y unos profesionales capaces de convertir dificultades en oportunidades y trabajo en pasión.

Gregorio, por partida doble, como mínimo: por brindarme esta gran oportunidad, y sobre todo, por la pasión que pones en tu docencia y que a tantos de nosotros ha llevado "al lado oscuro".

Compañeros de batalla, habéis hecho diferentes esas intensas noches de estudio, y especiales estos largos años de carrera. Hay un pedacito de cada uno de vosotros en este proyecto. Echad la vista atrás y fijáos en todo lo que hemos vivido y crecido juntos...

Ernesto y Marta, sois mis grandes puntos de referencia y de apoyo. Desde vuestra experiencia me habéis ayudado a construir la mía. Gracias.

Padres: me habéis dado todo. Hacer alusión a vuestro papel en mi etapa universitaria o en estos últimos meses sería desmerecer 26 años de sacrificio desinteresado por vuestra peque.

Los enanos del clan, capaces de sacarte una sonrisa en cuestión de segundos; Chus y Jero, mis hermanos adoptivos; Jorge, poca gente me entiende mejor.

Y a todas esas personas que han compartido conmigo cada momento de este intenso último año y que han puesto su granito de arena en este proyecto.



# Resumen

Los sistemas heredados son aún parte fundamental de la infraestructura de grandes compañías, asentados por su robustez, madurez y fiabilidad, fruto de un largo periodo de desarrollo.

Pero estos sistemas también están caracterizados por otro concepto: su coste. Altos costes de mantenimiento de los lenguajes heredados y de las plataformas subyacentes, en expertos, y en integración con nuevas tecnologías.

En este contexto, considerando tecnologías modernas como Java lo suficientemente maduras y consistentes para tomar el relevo, muchas compañías deciden migrar sus sistemas heredados a dichas tecnologías.

*itestra GmbH* es una consultora de software alemana especializada en la migración de sistemas heredados y enfocada hacia la eficiencia, optimización y reducción de costes. Como becario para esta empresa, he trabajado en el desarrollo de una herramienta de gestión de procesos cuyo fin es mejorar y adaptar las funcionalidades de las herramientas existentes en el área de la migración de plataformas, *itestra Task Manager*.

*itestra Task Manager* ha sido diseñada en forma de plug-in de Eclipse, escrita en Java y con persistencia a ficheros CSV. Las herramientas externas proceden de proyectos *open source*.

El presente documento expone los objetivos, metodologías y resultados de dicho trabajo de desarrollo.





# Summary

Legacy systems currently run successful companies. They are robust, mature and trustworthy, as a result of years of development.

Legacy is also identified by another concept: cost. High cost of maintenance of the legacy languages, high cost of the underlying platforms, high cost of experts, high costs of integration with newer technologies.

In this context, considering modern technologies such as Java mature and consistent enough as to take over, many companies are deciding to migrate their legacy systems to modern technologies.

Itestra GmbH is a German software consulting company specialized in the migration of these legacy systems and focused on efficiency, optimization and cost reduction.

As a working student for Itestra, I have worked on the development of a project management tool which goal is to improve the functionalities of the existing tools in the area of platform migration, *itestra Task Manager*.

*itestra Task Manager* has been designed as an Eclipse plug-in, written in Java and with CSV-file persistency. The external tools belong to *open source* projects.

The present document presents the goals, methodology and results of this development work.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Legacy technologies and systems . . . . .	1
1.2	itestra GmbH . . . . .	2
1.3	The migration process: from legacy to new technologies . . . . .	2
1.3.1	itestra's migration approach . . . . .	3
1.4	Struture of this document . . . . .	3
<b>2</b>	<b>Goals</b>	<b>5</b>
2.1	Problem description . . . . .	5
2.2	Main goal . . . . .	5
2.3	Secondary goals . . . . .	6
<b>3</b>	<b>State of the art</b>	<b>7</b>
3.1	The migration process . . . . .	7
3.2	Application Lifecycle Management tools . . . . .	9
3.2.1	Mylyn . . . . .	9
3.3	Technologies applied in the development of the tool . . . . .	10
3.3.1	Eclipse IDE . . . . .	11
3.3.2	File storage . . . . .	13
3.3.3	Application logging . . . . .	14
3.3.4	The Apache Software Foundation . . . . .	14
3.4	Stack Overflow . . . . .	15
<b>4</b>	<b>Design and implementation</b>	<b>17</b>
4.1	Requirements . . . . .	17

4.2	Appearance . . . . .	18
4.2.1	TaskList View . . . . .	19
4.2.2	Task Editor . . . . .	22
4.2.3	Updates Editor . . . . .	23
4.2.4	Preferences page . . . . .	25
4.3	Implementation . . . . .	26
4.3.1	Eclipse Extension points . . . . .	26
4.3.2	Data model . . . . .	27
4.3.3	Concurrency and synchronization . . . . .	29
4.3.4	Storage . . . . .	29
4.4	Methodology . . . . .	30
4.4.1	First analysis of requirements, study of the different alternatives and introduction to Eclipse plug-in development . . . . .	31
4.4.2	Attempt of integration with Mylyn . . . . .	33
4.4.3	First sketch of the tool . . . . .	35
4.4.4	Redesign and refactoring . . . . .	38
4.4.5	Scope redefinition . . . . .	42
4.4.6	Final development process . . . . .	42
<b>5</b>	<b>Conclusions</b>	<b>45</b>
5.1	Goals' accomplishment . . . . .	45
5.2	Application of learnt concepts and knowledge . . . . .	46
5.3	Acquired competencies . . . . .	47
	<b>Bibliography</b>	<b>51</b>

# List of Figures

3.1	Mylyn layout . . . . .	10
3.2	Eclipse IDE . . . . .	11
3.3	Eclipse SDK . . . . .	12
4.1	<i>itestra Task Manager</i> overview . . . . .	19
4.2	Task list view . . . . .	19
4.3	New task wizard . . . . .	20
4.4	Plug-in information dialog . . . . .	21
4.5	Active task in TaskList View . . . . .	21
4.6	Task editor . . . . .	23
4.7	Updates editor . . . . .	24
4.8	Plug-in preferences page . . . . .	26
4.9	First database schema of the plugin . . . . .	32
4.10	Mylyn's task view supporting the <i>itestra Task Manager</i> task . . . . .	34
4.11	First sketch of the tool layout on Eclipse . . . . .	35
4.12	Task structure as presented on the first implementation proposal . . . . .	37
4.13	Enhanced database schema . . . . .	40



# Chapter 1

## Introduction

This section introduces the project and the context in which it has taken place.

First of all, a short introduction on legacy systems, its main problems, and a perspective on the current situation will be presented, which is meant to set the background where the migration processes take place.

In this context, the software consulting company itestra GmbH specialised in migration of legacy systems will be introduced, including a brief description of its activity and methods, which specify the requirements for the development of such a tool as the exposed in this document.

Closing this section, the structure of this report will be briefly detailed.

### 1.1 Legacy technologies and systems

For the purpose of the *2008 National Survey on Legacy Systems and Modernization in the States* [1], the National Association of State Chief Information Officers (NASCIO) came up with this baseline definition of legacy system:

*A legacy system* is not solely defined by the age of IT systems (e.g. 20 years) as there are many systems that were designed for continued upgrades, but the term also focuses on elements such as **supportability**, **risk** and **agility**, including the availability of software and hardware support, and the ability to acquire either internal or outsourced staffing, equipment or technical support for the system in question. The term may also describe the system's inability to adequately support *line of*

*business* requirements or meet expectations for use of modern technologies, such as workflow, instant messaging and user interface.

Although the term *legacy* may suggest antique, old and currently uncommon situations, a significant number of companies nowadays stick with outdated operating systems, which run important and even critical applications, and might damage their ability to generate new business. Julian Dobbins, director of product management at Micro Focus, stated that every time a bank customer makes a money transfer, the transaction passes through a legacy platform [2].

The reasons not to migrate a legacy system consist mainly of the migration costs and the lack of liability of the new systems. But despite these inconveniences, staying outdated carries inefficiencies, uncompatibilities, security threats and significant costs of maintenance related to the lack of expertise and support available, which continuously decreases. These reasons make companies consider migrating their systems to new technologies, mostly not as a whole but in a progressive phased modernization.

## 1.2 itestra GmbH

itestra GmbH is a software service provider in the field of business-critical processes, systems and applications founded in Munich and expanded to Estonia, United Kingdom, USA and Spain.

Its service portfolio covers the whole software life-cycle from design, implementation and testing to putting into service, including migration of potentially existing systems and data, as well as consulting enterprises on effectiveness and efficiency of business processes and IT [3].

Having turned ten years old last summer, itestra GmbH is a young company which is carrying out an important amount of in-house projects to increase its resources.

## 1.3 The migration process: from legacy to new technologies

Despite of the reluctance of many companies to migrate their legacy systems to new ones, the high costs of maintenance, risks and lack of experts available for technologies becoming progressively older and more isolated is encouraging firms to accomplish these change processes. Anyhow, the complexity and costs of these processes set the limits to their speed and scope.



Different approaches exist to gradually and orderly migrate these systems, which will be exposed later (See section 3.1). In this section, itestra's method will be presented, aiming to serve as a foundation to the development of the Project Management Tool that this report refers to.

### 1.3.1 itestra's migration approach

itestra's migration approach consists of analysing the existing legacy source in order to extract business rules and use cases that can be documented, reviewed and reimplemented in a new technology with new architecture.

The main benefits of this approach are:

- It requires little time from business experts.
- It can be applied when no documentation and no experts for the existing business logic exist.
- It avoids the *moving target* problem, as the requirements can change only as fast as the legacy system changes.
- It allows to make use of all capabilities of the new platform –which is not possible, for example, when translating automatically.

## 1.4 Struture of this document

This document is divided in the following five chapters:

### 1. Introduction

Introduces the context for the design of *itestra Task Manager*: legacy systems and the migration process, itestra GmbH, and its migration approach as base for the development of the tool.

### 2. Goals

This chapter shows the objectives that have been pursued in this project.

### 3. State of the art

A review on the current situation on migration processes, other existing project management tools and the technologies which have been used for the development of the *itestra Task Manager*.

### 4. Design and implementation

On the steps taken to develop the *itestra Task Manager*.

### 5. Conclusions

A reflexion on the accomplishment of the goals, as well as on the influence of knowledge acquired in my degree that has helped on this project and on the competencies acquired by its execution. Finally, future lines of development are exposed.

# Chapter 2

## Goals

This section specifies and describes the objectives which lead the execution of this project.

### 2.1 Problem description

Migration projects are undertaken by developing teams in a context of changes and coexistence between old and new components. This makes some way to keep track of the individual progress of each of its components necessary, as well as of the modification of the initial requirements. Such a task is assigned to the team manager, and the resulting information is to be retrieved and shared in project and Application Lifecycle Management (ALM) applications. The market currently provides several products for this task, but the nature of these migration projects demands additional functionalities.

### 2.2 Main goal

The main goal of this project consists of the development of a project lifecycle management tool specifically designed for migration projects, to be incorporated to *itestra*'s in-house products.

It must be designed as an Eclipse plug-in, in order to be integrated with such Integrated Development Environment (IDE), fulfill the basic features of the existing tools and, in addition, allow the massive update of tasks selected upon one or several characteristic fields, and provide a workspace revision and a system of propagation of changes in the source files.

## 2.3 Secondary goals

The associated secondary goals can be divided into technical specific goals and general business and personal goals.

Regarding the technical goals, directly derived from the requirements of the project, the following are to be achieved:

- Deeper and wider knowledge and capabilities related to new technologies.
- Better understanding of the migration process.
- Competencies on database design and administration.
- Basis, goals, structure and behaviour of ALM tools.
- Specific knowledge about Eclipse IDE and plug-in development.

The goals related to business and personal aspects are set by the nature of the project, which will be implemented in a commercial environment and restricted by business limitations. Among them, we can list:

- Competencies on project administration and agile management.
- Information on values and strategies of business management.
- Improvement of language capabilities.

# Chapter 3

## State of the art

### 3.1 The migration process

Migrating an infrastructure consists of gradually substituting legacy components to new supported technologies. Given the magnitude of the systems and the costs of the process, it is a task that companies implement stepwise, in a context of coexistence of legacy and new components. The authors of the article *Legacy System Migration Approaches* [10] have classified migration processes as follow:

- Modernization

Modernization implies the application of changes in the existing system. This migration approach contains two main approaches:

- Direct migration

This approach considers the system a black-box, encapsulating its functionality and modernizing components such as its UI to make it more usable. These new components access the legacy components via a middle layer, in charge of the interoperation.

- Indirect migration

These methods imply reimplementing the legacy components through reengineering techniques.

- Hybrid approach

Methods in this category contain, in different ways, concepts and techniques of both

direct and indirect migration.

- Replacement

By means of engineering, its goal is to completely substitute one legacy system by another up-to-date system, implying source code rewriting, new architecture desing, usage of new technologies and new database schema definitions. Its three main approaches are:

- *Cold turkey*

In medicine, *cold turkey* is referred to the abrupt and complete withdrawal of a habit or addiction. In migration, it is the base of the replacement approach and consists of replace a legacy system by generating a whole new system from scratch, with new technologies running on a new platform. Both legacy and new systems must run simultaneously while migration is taking place.

This carries a huge failure risk among with other problems, which have contributed to the development of the other two replacement approaches.

- *Chicken little*

Performs the migration in eleven iterative steps, from the analysis of the logic of the legacy system and its division into modules to the generation of the new information system, in such a way that in case of failure of a step, only the failed step needs to be repeated.

Legacy and migrated systems can this way work in parallel, using a common gateway to the information, and the functionality is gradually migrated from one system to the other.

Nevertheless, this parallel work causes complex consistency issues. Such problems and the lack of guidance on testing during the migration process consist of its main counterparts.

- *Butterfly*

*Butterfly* emerges as a suitable answer to the problems of the two previous approaches. The migration process is divided in six main steps, and the new system will not be in production stage until migration is over.

## 3.2 Application Lifecycle Management tools

The Application Lifecycle Management (ALM) consists of the governance, development and maintenance of application software. For the sake of efficiency, this task is to be as automatized and simplified as possible.

ALM applications provide tools for managing and completing the phases of design, development, testing, deployment and ongoing enhancements, helping organizations to improve the way they design, build, test and adapt their software [11].

As a response to the increasing complexity of the software development task, an increasing number of ALM tools have appeared. The DZone ranking in 2009 cites the next as the most used:

- VersionOne Agile
- Aldon Lifecycle Manager
- ThoughtWorks Mingle
- Seapine TestTrack Suite
- AccuRev Suite
- TechExcel DevSuite
- Serena Dimensions 10
- Polarion Requirements
- IBM ClearCase and ClearQuest
- Microsoft Visual Studio Team System
- Rally Enterprise and Community Editions
- MicroFocus StarTeam

### 3.2.1 Mylyn

Not included in the list above, Mylyn is Eclipse's current ALM framework, in whose structure and design *itestra Task Manager* is primarily inspired.

It provides a tasklist tree view able to show, filter, order and group the list of tasks. The user can instantly see which tasks are high priority, completed or not in time; double clicking on the task, a task editor will be opened containing the information on the task to be checked or modified.

What is special about Mylyn is that every task has an associated context. By clicking on the circle on the left of a task, this task is activated, memorizing the resources used by the developer while performing its activity. Mylyn executes an algorithm to calculate the degree-of-interest of the workspace resources for such task, thus filtering the workspace contents and showing only the resources which it has learned are relevant for the task.

When activating a task, the most relevant resources will be opened in editors, and the package explorer contents will be filtered to show the rest of important files.

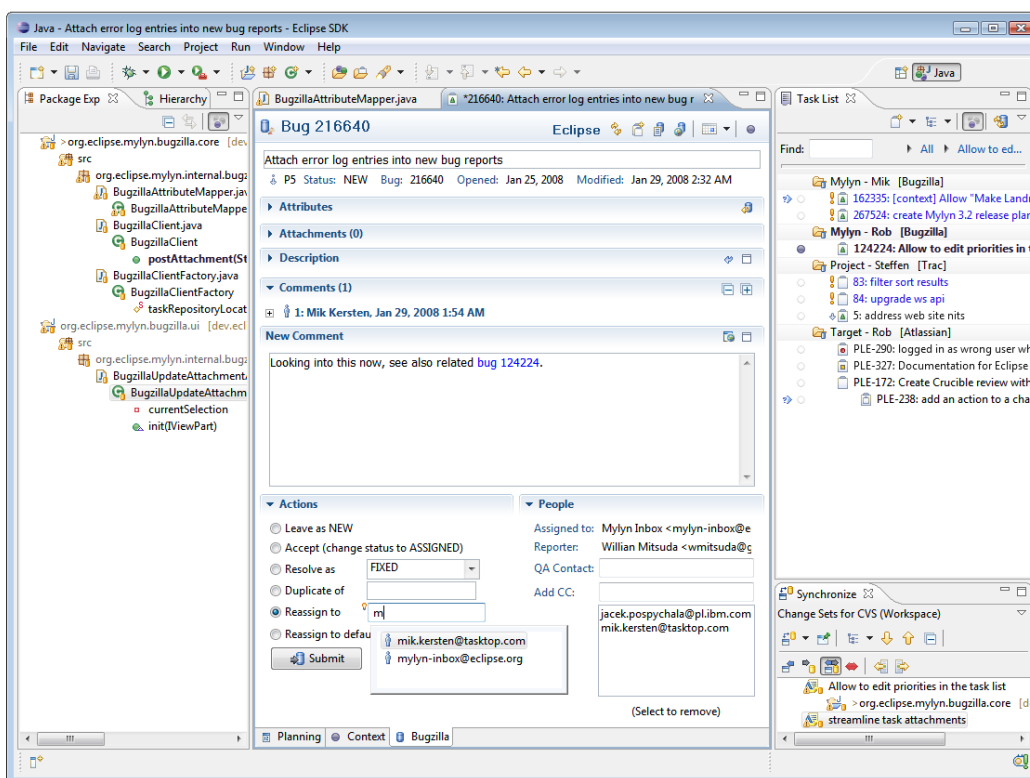


Figure 3.1: Mylyn layout

### 3.3 Technologies applied in the development of the tool

This section briefly introduces the technologies that have taken part in the development of the *itestra Task Manager*, starting from the target environment, Eclipse, and its plug-in development tools, and continuing with the tools that compose it.



### 3.3.1 Eclipse IDE

The Eclipse IDE is the Integrated Development Environment developed by the Eclipse Foundation, a not-for-profit, member supported corporation for development of open source software. Specially famous for its Java IDE –it has become the world’s most popular Java IDE [12]–, it provides IDEs for other languages such as C/C++, PHP or Python.

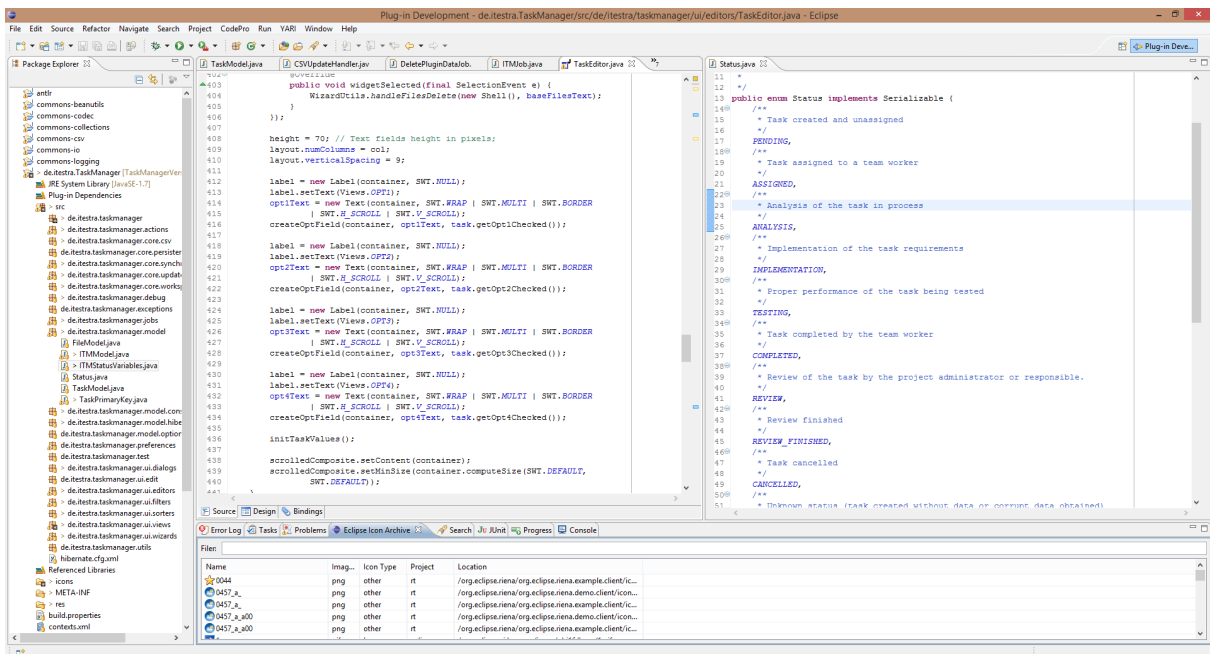


Figure 3.2: Eclipse IDE

Its more characteristic feature is its architecture: the Eclipse platform is structured around the concept of plug-ins. Eclipse presents a minimal core which provides connectors to allow extending its functionality.

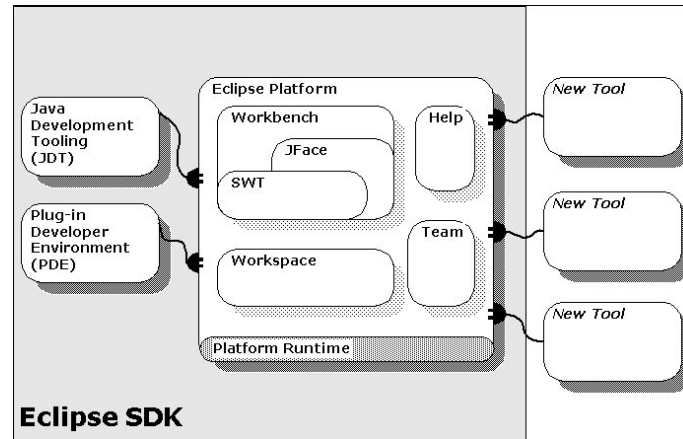


Figure 3.3: Eclipse SDK

As specified in Eclipse documentation,

”Plug-ins are structured bundles of code and/or data that contribute functionality to the system. Functionality can be contributed in the form of code libraries (Java classes with public API), platform extensions, or even documentation. Plug-ins can define extension points, well-defined places where other plug-ins can add functionality.” [13]

The *itestra Task Manager* contributes to some of these extension points to build its functionality on Eclipse (see 4.3.1).

### Eclipse PDE

It makes sense to think that, being plug-in based, the Eclipse Project provides the necessary tooling for contributors to develop plug-ins. This is the case of the Eclipse PDE project. According to the project definition,

”Eclipse Plug-in Development Environment provides tools to create, develop, test, debug, build and deploy Eclipse plug-ins, fragments, features, update sites and RCP products.” [16]

Due to its modularity, Eclipse provides the possibility to release several IDE distributions according to the end-user needs. For this development process, the most appropriate distribution

is Eclipse for RCP and RAP Developers. This package includes Eclipse PDE tools to integrate the plug-in into Eclipse.

Specially important in this tool integration within Eclipse are the libraries **.jface** and **swt**, which provide classes for handling many common UI programming tasks.

### 3.3.2 File storage

The plugin supports file import and export, using Comma-Separated Values (CSV) files. Importing data from a file can imply acquiring the whole list of tasks to save, as well as specific information to update in the existing tasks.

#### CSV files

Comma-Separated Values files store tabular data in plain-text form. Although it is a very common format, it has never been formally documented. An intent to standardize its structure can be found in the RFC-4180 on Common Format and MIME Type for Comma-Separated Values files [4], whose most important aspects regarding its use by the tool are:

- Each record is located on a separate line, delimited by a line break.

- Fields in a record are delimited by commas.

Other characters can also be used as field delimiters. For this reason, CSV is also known as Character-Separated Values.

- Optional header line.

Appearing as the first line of the file with the same format as normal record lines. It contains names corresponding to the fields in the records.

- Double quotes as escape characters.

Double quotes can be used to enclose fields, being its use necessary when the enclosed field contains reserved characters (commas -or field separators-, line breaks or double quotes).

### 3.3.3 Application logging

Logging consist of recording the events which happen while an application is running, by inserting log statements into the code. Many developers agree on its importance as a monitoring tool, which, properly stored and analyzed, provides valuable information on access patterns, code bugs, application failures and other aspects of the application performance and use, and may turn into the main source of information when diagnosing a problem.

Thus, many logging tools are being developed by communities and companies, such as the embedded Java Logger, *log4j* or *slf4j*, for Java.

For the development of *itestra Task Manager*, the chosen tool has been Apache's logger, *log4j*.

### 3.3.4 The Apache Software Foundation

The Apache Software Foundation defines itself as a community of developers and users. This American non-profit corporation supports the Apache software projects, currently more than 150 open-source "top-level projects"<sup>1</sup> covering a wide range of technologies [5].

The functionalities of two of these Apache top-level projects, Apache Commons and Apache Logging, have been used in the development of *itestra Task Manager*.

#### Apache Commons

Apache Commons [6] is a project focused on all aspects of reusable Java components, composed of three parts:

- **The Commons Proper**, a repository of reusable Java components.
- **The Commons Sandbox**, a workspace for Java component development.
- **The Commons Dormant**, a repository of components that are currently inactive.

Developers may freely join the Apache Community and contribute to these projects.

---

<sup>1</sup>*Top-level projects* are separated semi-autonomous areas into which the Apache software development is divided, some of them divided in sub-projects.

The Commons Proper is dedicated to the creation and maintenance of reusable Java components. Many useful utilities can be found in this section; the following have been used directly or indirectly in the development of the *itestra Task Manager*:

- BeanUtils

Wrappers around the Java reflection and introspection APIs.

- Codec

General encoding/decoding algorithms, such as phonetic, base64 or URL.

- Collections

Which extends the Java Collections Framework.

- CSV

For reading and writing Comma-Separated Values files.

- IO

I/O utilities.

- Logging

Wrapper around a variety of logging API implementations.

### Apache Logging

The Apache Logging Services Project [7] creates and maintains open-source software related to the logging of application behavior. It is divided in subprojects which cover different programming languages. *log4j* is the Apache logging library for Java, which has been used to handle the logging of the *itestra Task Manager*.

## 3.4 Stack Overflow

As specified in their website,

”Stack Overflow is a question and answer site for professional and enthusiast programmers.” [8]

This site is part of StackExchange Q&A communities [9] which currently hosts 129 communities focused on different topics. Its layout, highlighting the correct answers and making it easy for the user to navigate between related topics, its punctuation system, ranking questions and answers accordingly to their correctness and utility, its open membership configuration and its rewards system have contributed to make of this site a reference for every person who is stuck with any programming issue.

Consulting this site has solved most of the emerged problems and doubts during the whole development process.

# Chapter 4

## Design and implementation

May this chapter be considered the main part of this report, as it describes the learning and developing process itself.

The initial requirements of the *itestra Task Manager* will be presented firstly. The design and implementation of the resulting tool will be shown afterwards, to end up with the methodology followed and the steps taken to reach the final point.

### 4.1 Requirements

The *itestra Task Manager* requirements can be reduced to the general requirements of any ALM tool, plus extra specific requirements that make it useful for the specific migration environment. They are the following:

- Integrated into Eclipse IDE

The tool must be designed as an Eclipse plug-in, in order to be integrated and used from Eclipse IDE.

It must have access to the workspace resources, inspect them and open them when required.

- Creation, edition, visualization and removal of tasks

The tool must provide a way to create new tasks based on the workspace files.

It must also allow the user to modify the initial values of both a single task or a group of them.

The possibility to remove a task or group of tasks must be given as well.

- Model import and export

The user must be allowed to import a pre-existing list of tasks into the plug-in, as well as to export the current model.

- Back up

File back-ups of the model must be performed automatically and under user demand.

- Workspace inspection

The tool must be able to inspect the workspace, find the resources and add the selected ones to its data, as well as offer the user to create a new task set with these selected files.

- Workspace updates propagation

During the migration process, there will presumably be changes in the legacy code. The tool must provide a way of inspecting the workspace under user demand and, when found, alert the tasks related to the modified files of its update or removal. It may also offer the possibility to create new tasks for the new files encountered.

- Massive task update

The tool must allow the simultaneous modification of large numbers of tasks which fulfill determined conditions.

- User friendliness

The tool must be easy to use, simple and user-friendly.

## 4.2 Appearance

As already mentioned, the *itestra Task Manager* has been designed as an Eclipse plug-in. More specifically, it contains a TaskList View which allows the visualization and editing of the existing tasks, as well as general control of the tool; a preferences page to set its basic settings; a task editor which shows all the task information and allows its individual processing; and an



update editor to create update files, which will be used for a later massive task update. Several wizards have been implemented as well to assist the creation and modification of tasks.

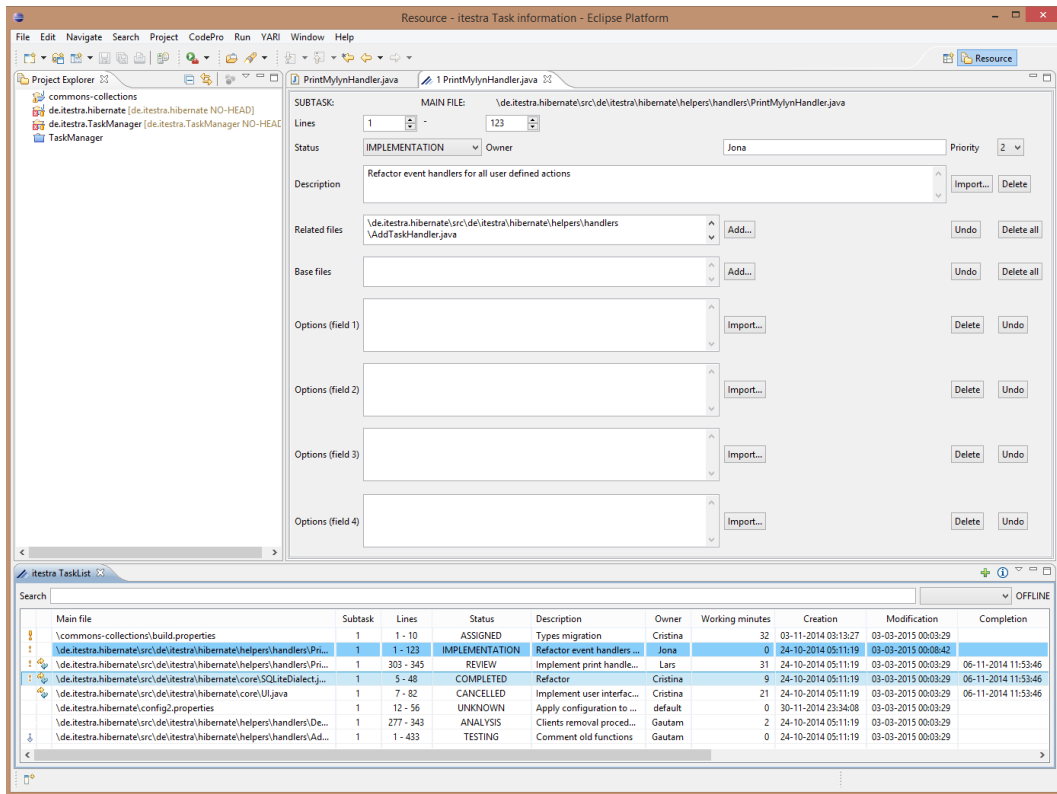


Figure 4.1: *itestra* Task Manager overview

### 4.2.1 TaskList View

This view contains a table listing all the tasks within the project, as well as the controls to manage the plug-in functionality.

Main file	Subtask	Lines	Status	Owner	Description	Creation	Modification
\commons-collections\build.properties	1	1 - 10	ASSIGNED	Cristina	Types migration	03-11-2014 03:13:27	30-11-2014 23:27:07
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\PrintMylynHandler.java	1	1 - 123	IMPLEMENTATION	Jona	Refactor event handlers for all user defined actions	24-10-2014 05:11:19	30-11-2014 23:20:00
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\PrintDBHandler.java	1	303 - 345	REVIEW	Lars	Implement print handler for partial con...	24-10-2014 05:11:19	30-11-2014 23:21:29
\de.itestra.hibernate\src\de\itestra\hibernate\core\SQLiteDatabase.java	1	5 - 48	COMPLETED	Cristina	Refactor	24-10-2014 05:11:19	30-11-2014 23:24:47
\de.itestra.hibernate\src\de\itestra\hibernate\core\UI.java	1	7 - 82	CANCELLED	Cristina	Implement user interface for booking a...	24-10-2014 05:11:19	30-11-2014 23:28:01
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\DeleteTableHandler.java	1	277 - 343	ANALYSIS	Gautam	Clients removal procedure	24-10-2014 05:11:19	30-11-2014 23:26:52
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\AddTaskHandler.java	1	1 - 433	TESTING	Gautam	Comment old functions	24-10-2014 05:11:19	30-11-2014 23:25:28

Figure 4.2: Task list view

- Model definition

The model can be created from the workspace. The plug-in inspects the workspace and presents the user the different file extensions so he can select which files to save, as well as whether to create a new task for each selected file.

The dropdown menu option *Inspect changes in workspace* will analyse the model files and look for updates or removals, as well as new files. It offers the possibility to spread the changes in the workspace to the model, which means notifying the tasks whose main, base or related files have been updated or deleted and create new tasks for the added files.

The task list can also be imported from an external file.

New tasks can be manually created using the "New task wizard".

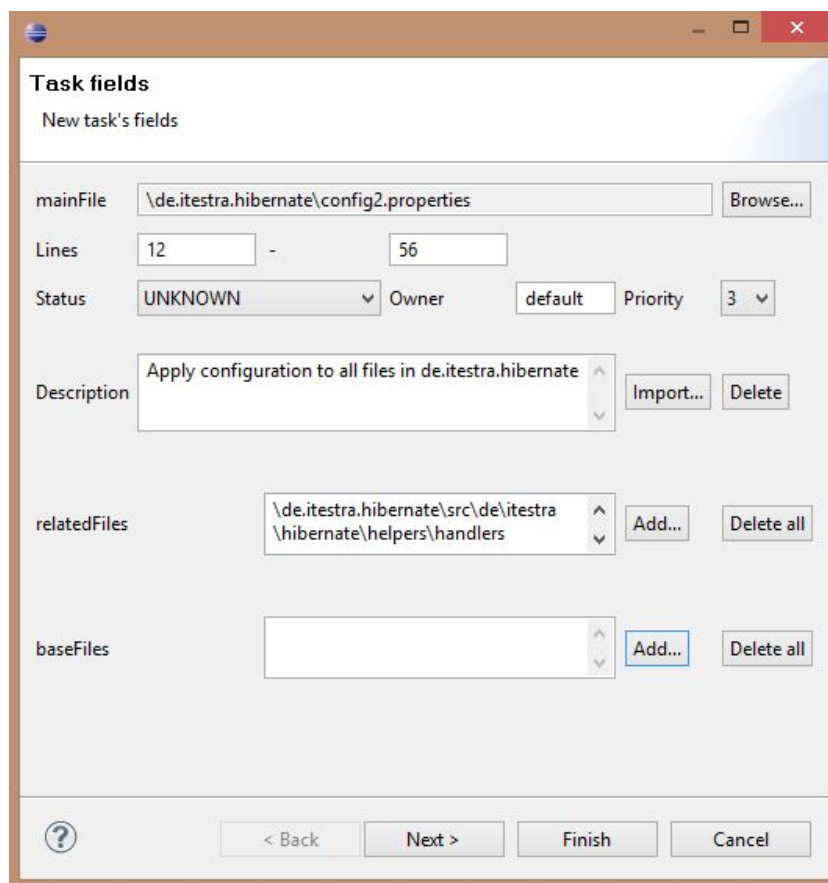


Figure 4.3: New task wizard

A task or group of tasks can be deleted from the model using the context menu.

The current state of the model can be shown when clicking the information button on the tool bar. It will prompt a dialog with plug-in related information, such as the number of

files and tasks contained in the model, the extension of the model files, and the date of the last workspace inspection.

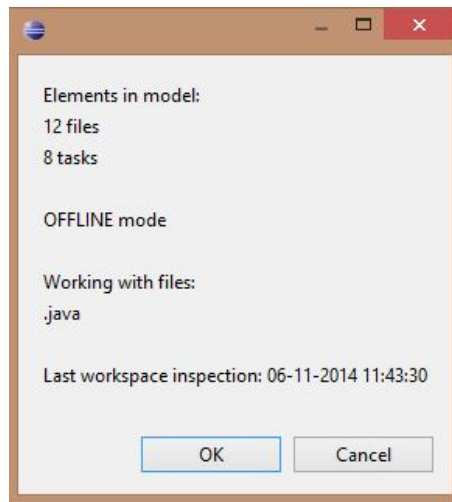


Figure 4.4: Plug-in information dialog

- Task activation and deactivation

By double clicking in one task, it will be activated: it will be highlighted in the view; all the editors will be closed to open the main file with a pointer to the beginning of the code block, and the task editor (see 4.2.2); the working minutes will be periodically updated, and the previous active task, if existing, will be deactivated. Double clicking on an active task will deactivate it without activating any other.

Main file	Subtask	Lines	Status	Owner	Description	Creation	Modification
\commons-collections\build.properties	1	1 - 10	ASSIGNED	Cristina	Types migration	03-11-2014 03:13:27	30-11-2014 23:27:07
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\PrintMyIhmHandler.java	1	1 - 123	IMPLEMENTATION	Jona		24-10-2014 05:11:19	30-11-2014 23:20:00
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\PrintDBHandler.java	1	303 - 345	REVIEW	Lais	Implement print handler for partial con...	24-10-2014 05:11:19	30-11-2014 23:21:29
\de.itestra.hibernate\src\de\itestra\hibernate\core\SQLiteDialect.java	1	5 - 48	COMPLETED	Cristina	Refactor	24-10-2014 05:11:19	30-11-2014 23:24:47
\de.itestra.hibernate\src\de\itestra\hibernate\core\UI.java	1	7 - 82	CANCELLED	Cristina	Implement user interface for booking a...	24-10-2014 05:11:19	30-11-2014 23:28:01
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\DeleteTableHandler.java	1	277 - 343	ANALYSIS	Gautam	Clients removal procedure	24-10-2014 05:11:19	30-11-2014 23:26:52
\de.itestra.hibernate\src\de\itestra\hibernate\helpers\handlers\AddTaskHandler.java	1	1 - 433	TESTING	Gautam	Comment old functions	24-10-2014 05:11:19	30-11-2014 23:25:28

Figure 4.5: Active task in TaskList View

- Task editing

Tasks can be individually and collectively modified.

- The task status can be modified directly by clicking on its field.

- The context menu provides an option to fast-edit the status, priority or owner of one or several tasks at a time.
  - It also offers a more complete edition method through wizards, for a task or a set of tasks. Fields such as the main file or creation date are not modifiable, some others are automatically but not manually editable, such as the modification and completion dates or the subtask.
  - The active task can be edited via the task editor (see 4.2.2).
  - A set of tasks which fulfill certain conditions can be collectively updated via a CSV file, which can be created using the updates editor (see 4.2.3).
- 
- **Sorting and filtering**

The task list can be filtered using the search box and combo at the top of the table, and sorted by any of its fields clicking on its column header.

The order of the columns can also be modified dragging and dropping their headers.

## **4.2.2 Task Editor**

Presents the information related to the selected task, and offers the user the possibility to modify its fields.

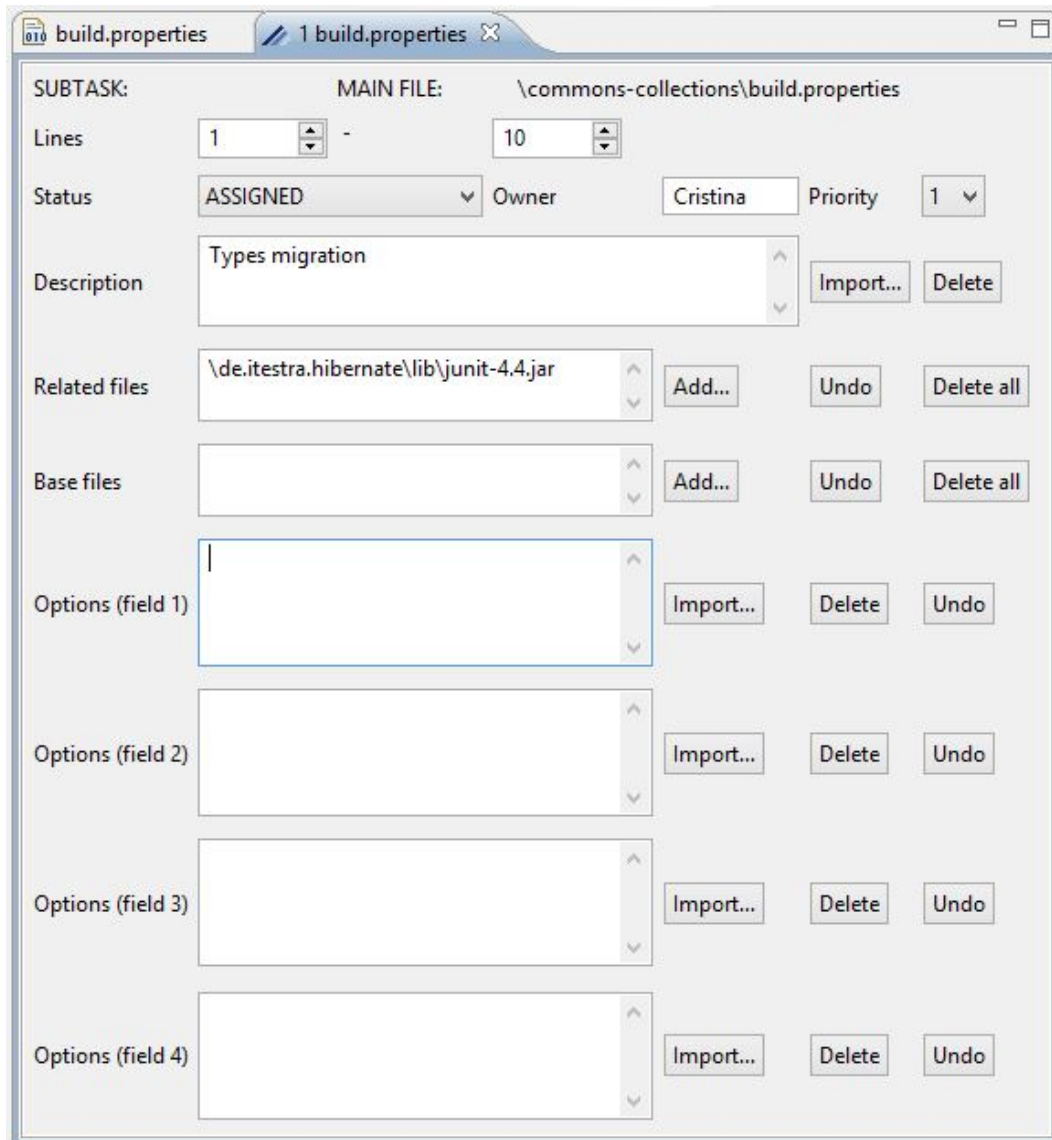


Figure 4.6: Task editor

### 4.2.3 Updates Editor

The Updates editor provides a graphic way of configuring the parameters of the tasks which must be updated, as well as the modifications to be done in such tasks.

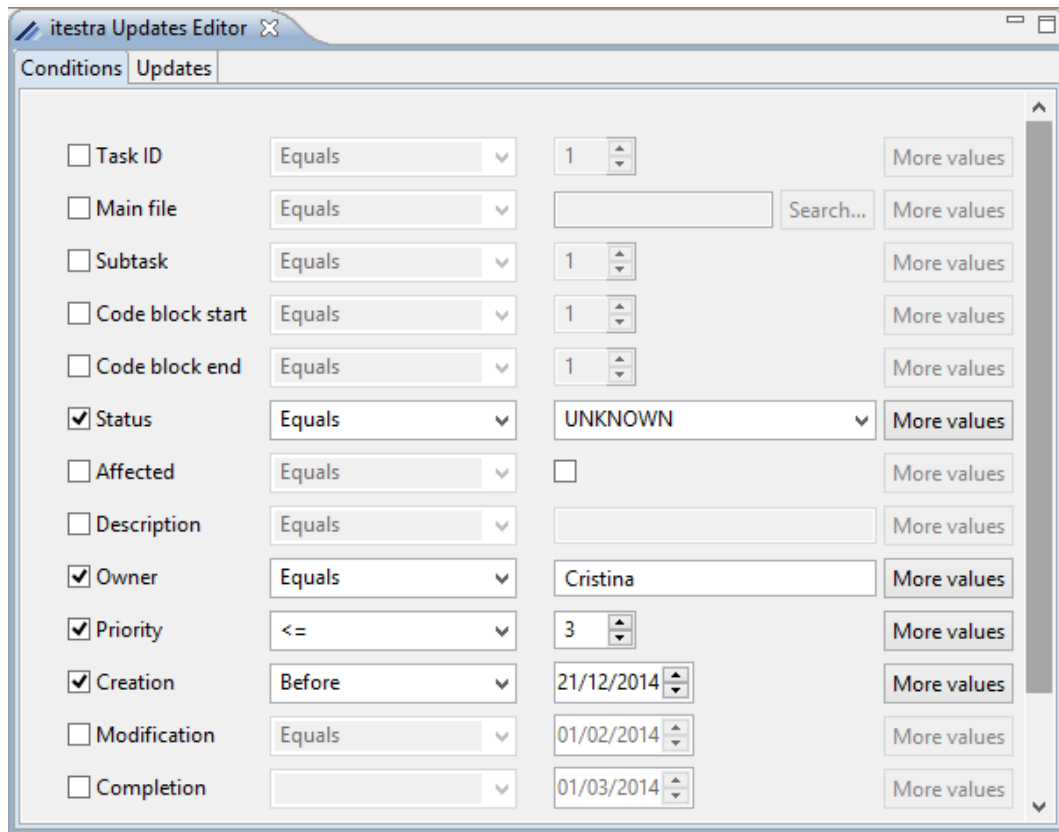


Figure 4.7: Updates editor

It contains two tabs from which to select both the conditions to look for compatible tasks and the updates to be performed, and offers different selection and updates options, which are, for conditions:

- Equals
- Not equal to
- Greater than
- Greater than or equal to
- Less than
- Less than or equal to
- After
- Before
- Between
- On day
- On week

- On month
- On year
- Contain
- Not contain
- Equals case sensitive
- Not equal to case sensitive
- Contain case sensitive
- Not contain case sensitive.

And for updates:

- Set
- Delete
- Add

The conditions admit several values to be searched for, and are constrained by the restrictions in types (i.e., Condition *On day* can only be applied for dates, as well as *Between* can not be applied to strings). Such restrictions apply also to the updates, which may perform different behaviours depending on types (i.e., *Add* will add a string at the end of a string field, but will save the addition of two integer values).

#### 4.2.4 Preferences page

The plugin has its own preferences page in Eclipse preferences dialog, from where general settings regarding its behaviour can be changed.

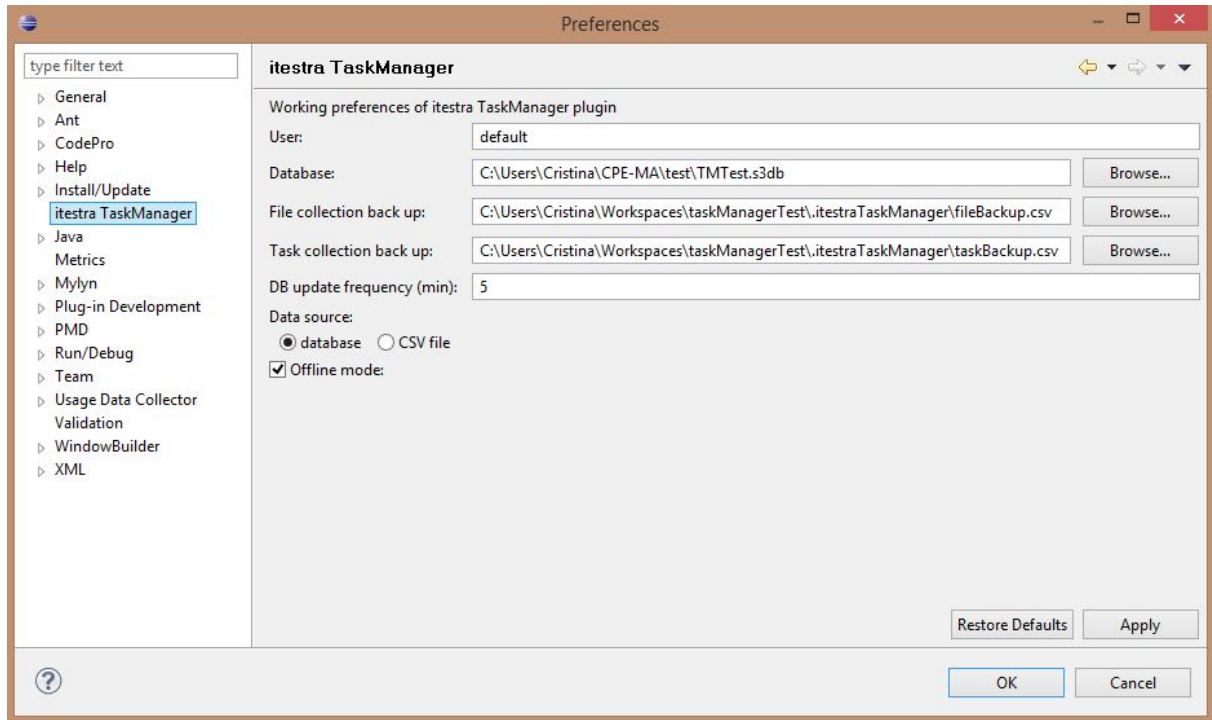


Figure 4.8: Plug-in preferences page

## 4.3 Implementation

### 4.3.1 Eclipse Extension points

As referred to in section 3.3.1, the Eclipse plug-in structure allows plug-ins to extend its connectors in order to add functionalities. The functionalities provided by the *itestra Task Manager* contribute to the following Eclipse extension points:

#### **org.eclipse.ui.preferencesPages and org.eclipse.ui.preferences**

Provide the tools required to add items to Eclipse preferences and a page to edit these items in Eclipse preferences dialog.

#### **org.eclipse.ui.views and org.eclipse.ui.perspectiveExtensions**

Extended to create the itestra TaskList View, included in the existing Eclipse Java perspective.



## **org.eclipse.ui.editors**

Supports the implementation of both Task and Updates Editors.

### **4.3.2 Data model**

The data model consists of two different models: the file model storing the information about the files considered in the tasks, and the task model including all the tasks in the project. Files can be related with tasks in three different ways:

- **Main file**

The main file of a task is the file in which the activity of the task is focused. Generally, it is a file belonging to the legacy source code to migrate.

- **Base files set**

Set of files which highly influence the task, usually belonging to the legacy source code too. They complement the main file.

- **Related files set**

This set contains files, either in the legacy or modern project, that are somehow related to the task. It is a decision of the manager project –or user– to decide the nature and scope of such relation.

#### **File model**

The file model contains a map with information about the files which are relevant for the migration project, and the functions to access, modify and control it. The files must be instances of IFile objects, this means, files contained in the Eclipse workspace.

The file information is stored in File objects, whose structure is:

- **File path:** Path of the resource regarding the workspace root.
- **Checksum:** MD5 of the file content, saved to control the changes in the file through progressive updates of the source project.
- **Modification date** when the file information was last updated.
- **Main tasks set:** Set of all the tasks of which the resource is main file.

- **Base tasks set:** Set of all the tasks of which the resource is base file.
- **Related tasks set:** Set of all the tasks of which the resource is related file.

The file map contains File instances ordered by their file path.

### Task model

The task model is the counterpart of the file model for tasks, containing the collection of tasks and their associated methods. A task contains the following information:

- **Main file:** The main file on which the task's job is based.
- **Subtask:** A file can be scheduled to be migrated in different tasks. This means that different tasks can be assigned to the same main file. This field is a numeric value used to difference between tasks assigned to the same main file.
- **Code block:** Two numeric fields store the lines in the main file which contain the code relevant to the task.
- **Status:** The phase in which the task is in the project. From its creation to its completion, the task can go through this set of status:
  - Pending
  - Assigned
  - Analysis
  - Implementation
  - Testing
  - Completed
  - Review
  - Review finished
  - Cancelled
  - Unknown
- **Affected:** The legacy code can –and most probably will– suffer changes during the whole migration process to fulfill new requirements of functionality. This flag alerts when a relevant file for the development of the job assigned to the task has been updated.
- **Set of affecting files:** Contains the paths of the files related to the task that have been updated or deleted.

- **Description:** Contains a detailed description of the task to be performed.
- **Owner:** developer in charge of the task.
- **Priority** of the task in the project context, from 1 –the highest– to 5 –the lowest.
- **Creation date** of the task
- **Modification date:** the last moment when the task has been modified
- **Completion date:** the completion date, if completed.
- **Working minutes:** amount of time invested in the completion of the task.
- **Last modified by:** developer who modified the task the last time.
- **Base files set:** collection of files which influence the task directly, normally belonging to the legacy set.
- **Related files set:** set of files related to the task.
- **Options:** four fields to be filled in according to the characteristics and needs of the project.

These Task objects are stored in a task map, ordered by their main file and subtask. Another auxiliary map stores all the subtasks related to a same main file, to provide direct access to them without needing to inspect the whole task map. A TaskPrimaryKey object containing such main file and subtask number is used for such classification.

### 4.3.3 Concurrency and synchronization

Eclipse provides the Jobs framework, which allows multiple operations to be run in the background and provides feedback of the job status to the Eclipse platform. The plugin functionality has been implemented through such Job objects.

To avoid concurrency problems, such jobs access the plug-in logic through a DataStorage singleton, which contains synchronized methods and works as synchronization tool to access model and functions.

### 4.3.4 Storage

A back-up of the model is stored in the workspace –or another location selected by the user– when closing Eclipse or the plug-in, performing certain operations, as set-up or important model changes, or on user demand.

This back-up is stored in a CSV file, containing one task per line,

If existing, this file is read on plug-in start, being its tasks imported. When importing, the file model will be restored from the information contained in the tasks and workspace: the CSV file with tasks contains the file paths, which will be analysed to retrieve the remaining information.

The format of the CSV lines is specified in the next schema:

```
<taskId>,<mainFilePath>,<subtask>,<codeBlockStart>,<codeBlockEnd>,<status>,<affected>,<description>,<owner>,<priority>,<creationDate>,<modificationDate>,<completionDate>,<workingMinutes>,<baseFilePaths>,<relatedFilePaths>,<affectingFilePaths>,<lastModifiedBy>,<opt1>,<opt2>,<opt3>,<opt4>
```

The plug-in handles empty values by setting the task properties to default, so it is not necessary to specify all the task fields. The following is an example of CSV input for a task:

```
,\TMClient\src\view\AbstractController.java,1,56,138,IMPLEMENTATION,false,
"Fix controllers hierarchy",Ismael,3,22-10-2014 15:12:48,27-10-2014 12:15:21,,68,
\TMClient\src\view\PersonController.java&&\TMClient\src\view\ContractController.java,
,,Cristina,,,
```

## 4.4 Methodology

Let this section begin with an important consideration about my learning method: I learn by doing. This means trying works better than reading in my learning process. This has conditioned the development methodology of the project.

The development process can be classified in the following milestones:

1. First analysis of requirements, study of the different alternatives and introduction to Eclipse plug-in development.
2. Attempt of integration with Mylyn.
3. First sketch of the tool.
4. Redesign and refactoring.
5. Scope redefinition.
6. Final development process.

During such process, three important modifications on the basic concepts of the plug-in have been undertaken:

1. Integration with Mylyn discarded
2. Data model structure modified
3. Database storage discarded

#### **4.4.1 First analysis of requirements, study of the different alternatives and introduction to Eclipse plug-in development**

This initial phase is basically devoted to study the requirements and gather information about everything related to the tool.

A first set of conversations about the tool takes place with Jonathan Streit, head of R&D, in which the background, context and requirements are introduced.

- itestra takes care of the whole process of migration, from the analysis of the existing systems to the development and set on production of the new ones. Several tools are used for the analysis of the legacy systems. They provide results on code structure and quality that can be used as start set to create the tasks in which to structure the project. These tasks need to be introduced into the application to be developed.
- As Eclipse is the main IDE used in the migration process, this tool must be integrated in it as Eclipse plug-in, so it can have access to Eclipse resources and be used as a part of it, not needing additional external tools. It must manage the resources in the tasks so they can be analysed after changes and opened in their appropriate Eclipse editors.
- Once the developer has integrated this list into the Eclipse environment, he must be able to modify their contents so they provide a faithful image of the status of the processes.
- As the legacy system may change between code updates, the initial tasks may also suffer important modifications that need to be notified to the tool. These modifications may present common elements for a set of tasks, so this modification process needs to be automated and able to edit tasks massively.
- Last but not least, it must be user-friendly: easy to learn and easy to use.

With this starting requirements, Lars Lehmann, developer, and I defined a first list of characteristics.

- Eclipse and Java version

To ensure compatibility with the project it was first conceived to, the tool would be developed using Java 1.6 and Eclipse 3.7 (Indigo).

- Database storage

The tasks would be stored in a central database to which all developers will have access.

SQLite was chosen as database management system, and as persistence framework we decided to use EclipseLink, but had to switch to Hibernate due to bugs and incompatibilities with the Eclipse plug-in development.

The first database schema consisted of a unique table as follows:




TASK		
	filePath	TEXT
	fileName	TEXT
	subtask	INTEGER
	codeBlockStart	INTEGER
	codeBlockEnd	INTEGER
	status	TEXT
	description	VARCHAR
	owner	TEXT
	priority	INTEGER
	creationDate	TIMESTAMP
	modificationDate	TIMESTAMP
	completionDate	TIMESTAMP

Figure 4.9: First database schema of the plugin

- For offline work, file persistence to CSV files would be implemented. Apache Commons CSV would be used to manage such operations.
- Mylyn framework would be extended to support our tasks version. Mylyn task elements would be used to store our task elements and its views and actions would host our tool functionality.

## 4.4.2 Attempt of integration with Mylyn

### Mylyn analysis

The Mylyn project provides extensive information for integrators. The Mylyn integrator reference<sup>1</sup> presents the Mylyn structure focusing on integration perspectives. This is the basic document every developer should take into consideration when building on Mylyn, and the first step to take.

This wiki details the functionality and extensibility of the Mylyn modules. Let's take a brief look at its structure.

Mylyn distinguishes between internal and Application Programming Interface (API) components. API components are intended to be extended by external developers and so, kept stable to ensure maintainability of the extensions, while internal components are subjects of changes which may affect the integration. Thus, API components are the ones meant to be extended while the use of internal components is discouraged.

The tool is developed extending the following APIs:

- Commons API

Provides common facilities used by the other Mylyn components.

- Tasks API

Contains all the functionality related to the tasks: task list, editors, views and repositories.

- Monitor API

Module used for collecting information about the user's activity in Eclipse, to create a degree-of-interest model used to filter the context elements in which is the base functionality of Mylyn: reduce the workspace contents to the relevant information about the selected task.

- Context API

Contains all the components related to filtering, expanding and decorating Eclipse viewers and editors.

---

<sup>1</sup>[wiki.eclipse.org/index.php/Mylyn/Integrator\\_Reference](http://wiki.eclipse.org/index.php/Mylyn/Integrator_Reference)

### Discarding Mylyn integration as an option

After working on Mylyn integration for some time, compatibility and performance issues were found that questioned the convenience of such integration:

- Extending Mylyn to fulfill the requirements of the tool to be developed implied accessing and modifying internal components, threatening the tool's maintainability.
- The developing process focused on attempting to adapt the tool to Mylyn, instead of adapting Mylyn to the tool. Mylyn introduced strict conditions to the development possibilities.
- Adapting the task structure to Mylyn's task structure implied losing control of the task modifications (i.e., some fields which were meant not be modified were set to text fields which the user could freely change in Mylyn's structure), which could lead to performance problems. The user was free to perform operations that would compromise the tool's functionality.

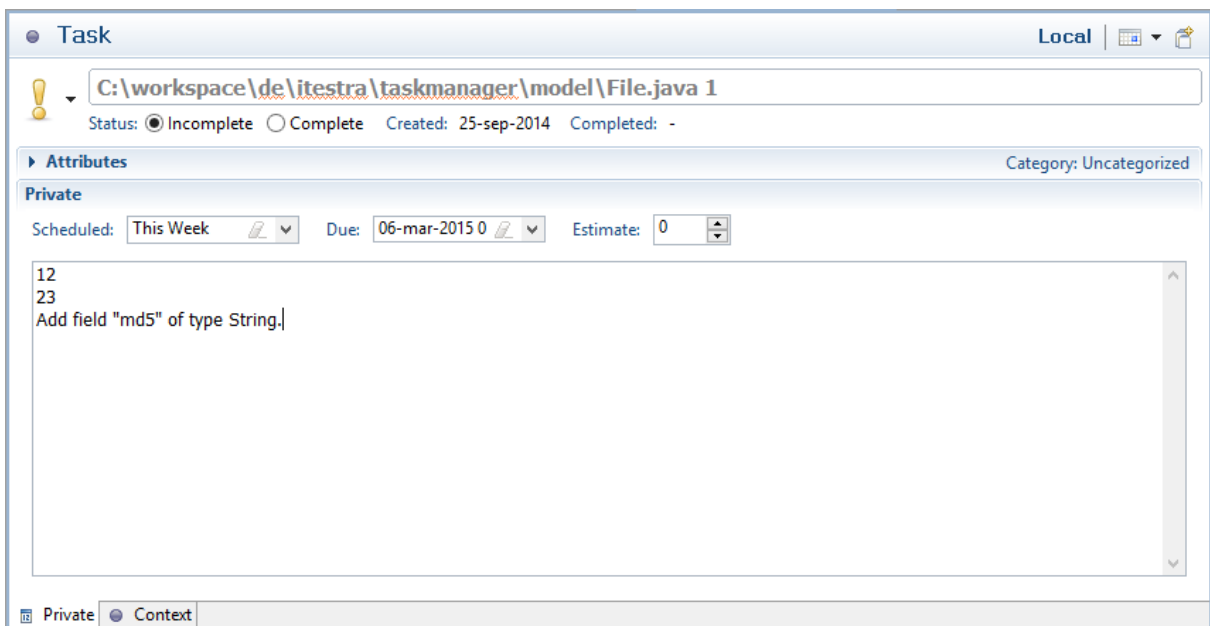


Figure 4.10: Mylyn's task view supporting the *itestra Task Manager* task

- Mylyn's approach is not the best match for our approach.

These reasons made us consider that having to adapt the whole tool to Mylyn was not a good approach. Some Mylyn components were useful and could be imported to the tool, but adapting



the whole tool to Mylyn just to take advantage of such components was not worthy. Instead, we decided to analyse them and add them to an independent plug-in.

Despite discarding Mylyn as an alternative, its inspection and the integration process worked as a guide into Eclipse plug-in development, so at the end of this step I had already learnt the basis of Eclipse integration and plug-in development using Mylyn as a practical case.

### 4.4.3 First sketch of the tool

After deciding not to build the tool on Mylyn, and based on Mylyn's layout in Eclipse, a first proposal of the tool design and implementation is presented and developed.

The proposed layout includes a menu in Eclipse menu bar, a tasklist view on one side of eclipse, and a task view on the bottom, as shown in the following sketch:

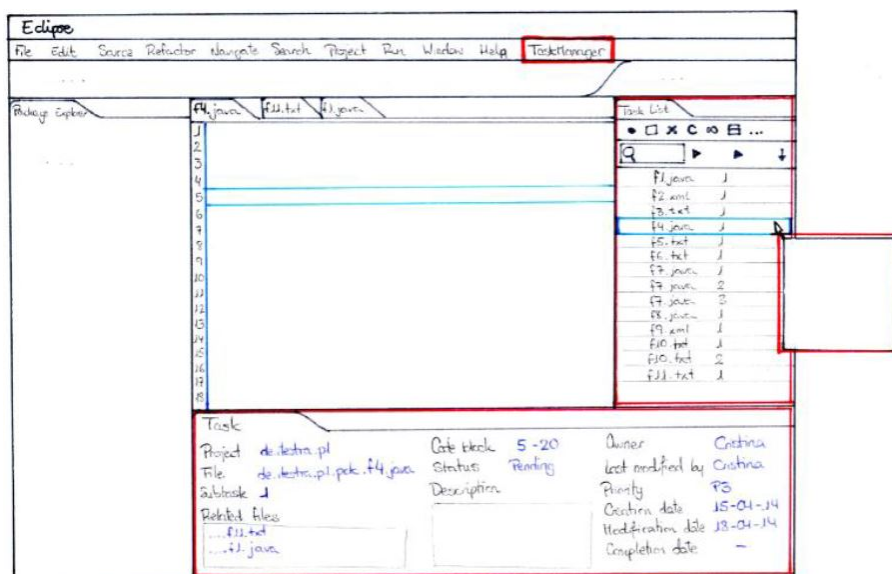


Figure 4.11: First sketch of the tool layout on Eclipse

More specifically:

- Menu in menu bar

Contains the general plug-in actions, such as import/export and configuration.

- Views

Contain the plug-in functionality. Two views are proposed:

– TaskList View

Contains:

\* Task list

Table viewer containing all the tasks, with resizable and draggable columns to adapt to the user needs of information.

It also provides sorting and filtering mechanisms to reduce the visual load and order the tasks by the most significant parameter at each time.

\* Tool bar

With the main commands to handle the task list, such as synchronization, creation of new tasks, and rest of actions performed over the task list as a whole or significant parts of it.

\* Context menu

Allows the user to perform actions affecting only the selected task or group of tasks, such as editing, activating, opening...

– Task View

It contains the task information and offers appropriate edition tools for the modifiable fields.

• Editors

Eclipse default editors would be used to open the task files when required, with the pointer on the first line of the code block to migrate.

The task structure is also modified, including new fields, and restricting the user actions on them:

• Limited edition values are admitted.

• Some modifications are performed automatically as response to other actions –i.e., completion date is set when the status is set to Complete– so the user does not need to take care of such fields.

• Editing the fields which characterize the task is not allowed.

The proposed task structure is as follows:

FIELD NAME	DESCRIPTION	VALUES	MODIFIABLE	SPECIFICS
File path	Project	String	NO	Must be in the workspace
File name	Path to the file within a project. It will be opened in an editor when selecting/activating the task	String	NO	Must be in the workspace
Subtask	Identifies different tasks concerning a same file	Integer > 0	NO	
Code-block start	Start point of the task. Line where the editor cursor will be set	Integer > 0	-	
Code-block end	End point of the task	Integer > code-block start	-	
Status	Current status of the task (Pending, Assigned, In process, Completed...)	String. Delimited.	YES	
Description	Detailed description of the task	String	-	
Owner	Person to whom the task has been assigned	String	-	
Priority	Priority of the task in the project	Integer [1 -5]	-	
Creation date	Date of creation of the task	Date	NO	Set in the DB or CSV. If not specified, it will take as value the import date.
Modification date	Last date of modification of the contents of the task	Date	AUTO	Automatically set when a field of the task is modified.
Completion date	Date when the task is completed	Date	AUTO	Automatically set when the status is changed to Completed.
Last modified by	Person who last modified the task contents. Useful for tracking	String	AUTO	Automatically set when a field of the task is modified.
Related files	Important files for the task. Optionally opened in editors as well as the main file	String	-	
Influenced files	Files affected by this file	String	-	Used for spread of notification of relevant changes.
Optional field 1		String	YES	Additional files to store information dependent on the different migration projects.
Optional field 2		String	YES	
Optional field 3		String	YES	
Optional field 4		String	YES	

	Key parameters
	Proposed new fields
	Fields required by Jona
	- Not decided

Figure 4.12: Task structure as presented on the first implementation proposal

The developing cycle is divided in two main milestones:

### 1. Simple UI

A first release would be a simple implementation of the basic functionalities: import and export of the task list from CSV and database, and primitive views and actions.

- Basic functionality.
- Menu, Tasklist view with context menu and Task view.
- Open files in their appropriate editors.
- Manual modification of the task fields.
- Manual synchronization.
- No advanced functionality, such as automatic modification of the fields, filtering, spread of relevant changes, automatic synchronization, etc.
- Working basis for the next step.

## 2. Advanced functionality

During this step, advanced functionality would be gradually included, prioritizing in the process, until getting a full running version of the plug-in.

These steps were not strictly followed: some requirements reserved for stage 2 were easy to implement so they were implemented in stage 1. After stage 1 was developed, the tool was presented to the head of R&D, who presented some modification requests. This brings us to the next step.

### 4.4.4 Redesign and refactoring

After inspecting this first, still uncomplete version of the plug-in, the head of R&D, Jonathan Streit, asked for several changes in its structure and design.

- To add all the functionality of the plug-in in the views, without including a menu in the menu bar. As many plug-ins tend to add such menus, the menu bar becomes crowded, so the request is to avoid it if possible.

A preferences page will be added to host the plug-in configuration options, and the general options will be included in the Tasklist view tool bar or menu.

- To improve the database schema, as follows:
  - Better schema with separate tables for files and tasks to avoid redundancy by making use of one-to-many and many-to-many relationships.
  - Create an object File, containing valuable information of the files instead of only their paths, and change the fields containing several file paths from String to Set of File objects.
  - Use enums for fields with a restricted set of values, such as the Status.
- Inversion of the logic for spreading the file system changes.

This process was proposed as follows:

The field "Influenced files" contains the files on which the contents of the task have influence. When the user considers that an operation regarding such task is relevant for this

set of files, he can select the option "Spread relevant change" in the task context menu, and all the tasks whose files are included in the set of influenced files will be notified by changing its status, even when having been set to complete. To ease the tracking, fields "Modification date" and "Last modified by" may also be updated.

This approach is discarded and changed by this other:

The plug-in must be able to perform an analysis of the files in the workspace and find the differences between the actual version of the files and the stored one, either comparing the last modification date or performing a checksum check. When a file has been either deleted from the workspace or updated, the plug-in must locate all the tasks which content such file and notify them about the changes occurred –whether the file has been updated or deleted, and how the file affects the task (if it is its main file, or one of its related or base files).

- Upgrade to Java 1.7.

These specifications set the base of the final version. Schematically,

- Changes in the database
  - Database schema modified

From the single table schema to a schema with two entities, Task and File, related as follows:

    - \* 1:n relationship Task - File through field mainFile
    - \* n:m relationship Task - File through field relatedFiles
    - \* n:m relationship Task - File through field baseFiles
  - Primary key changed from natural to surrogate, to avoid composed primary keys and detach the technical development from the business logic.

The next figure shows the resulting DB schema:



– CSV mass update

The update information is read from a CSV file with the following structure:

```
<task-values-to-search>,<task-values-to-update>
```

Where

```
<task-values-to-search>
```

is a serie of 22 comma-separated values corresponding to the 22 task fields which structure is

```
<filter-mode> <reference-values>
```

And

```
<task-values-to-update>
```

is a second serie of 22 comma-separated values with the updates to perform to the tasks that meet the requirements of the first serie. Its structure is analog to the first part of the line,

```
<update-mode> <new-value>
```

All fields are optional, and empty fields will not be filtered. This means that a set of empty conditions will update all the tasks in the model, as well as an empty set of updates will not perform any change in the selected tasks.<sup>2</sup>

The update operation is secuential: the second update will be performed after the first one. Thus, when providing different updates in the same file, the developer must pay attention to the order, as undesired conditions may occur.

For example, this CSV file:

```
,,,,,, = PENDING,,,,,,,,,,,,, SET ANALYSIS,,,,,  
,,,,,, = ANALYSIS,,,,,,,,,,,,, SET PENDING,,,,,
```

would first change all the tasks in PENDING status to ANALYSIS, and then change them back to PENDING, as they will evaluate true for the second condition, with a global result of no updates.

---

<sup>2</sup>The filter and update modes refer to the options explained in 4.2.3

- Updates editor

This editor provides a graphic interface for the user to create mass updates CSV files, containing only one update line.

#### **4.4.5 Scope redefinition**

Diverse design aspects, specially regarding online-offline mode availability, carried technical problems regarding synchronization with the database that could not be solved. After several time consuming attempts and approaches which proved wrong and given the advanced state of the project and the relative importance of such an aspect in the whole project, the decision was made to postpone the database access to a further iteration and focus on the development of a self dependent file-based ALM tool.

#### **4.4.6 Final development process**

After a learning process and two major changes in the plug-in structure, this final stage is based on the requirements previously defined in section 4.1 and its result is the one shown in sections 4.2 and 4.3. The functionality was added in progressive iterations from the basics to the fine tunings until obtaining the final version.

- Model adjustment to offline mode

Hibernate manages the relations between the entities. Without Hibernate, such relations need to be included in the model processing, so both task and file models keep up-to-date information on the relations between them.

- Task modification

Add fields containing the record of influencing files. This information was shown before in the description, but an independent field makes its tracing and processing more efficient.

- Computation of the time spent in the implementation of the different tasks
- Modify UI elements to improve their usability



For instance, the kind and characteristics of the file selection dialogs, or the contents of the information prompts.

The labels' content is subject of consideration as well, searching intuitive terms for the user to replace the currently technically influenced ones.

- Debug and testing

Find hidden bugs trying to cover all suitable situations via testing, and solve them.

- Beauty shop

Improve the esthetics of the plug-in UI.

After this final step, the *itestra Task Manager* is finally ready to use. It is still subject of being tested in real life projects where real processes need to be managed.



# Chapter 5

## Conclusions

This closing chapter evaluates the accomplishment of the established goals and reflects on how knowledge acquired during the degree has contributed to this work and how it has been extended with newly acquired competencies derived from it.

### 5.1 Goals' accomplishment

After nine months of development, finally the proposed ALM tool is ready for testing, use and expansion, providing new functionalities not offered in other similar tools which are meant to assist the particular development process of system's migration.

The technical goals related to such work have been accomplished in a high rate, still being subject of further improvements and contributions. I have improved significantly my technical skills regarding design and implementation of features.

I possess general knowledge of many modern technologies and deeper insights in topics such as plug-in development for Eclipse, database design and administration, ALM tools design, etc.

My work has been focused towards migration, which has given me the chance to start learning such processes.

Being responsible at a high rate of my time scheduling has provided significant opportunities to improve my planning methods.

Working at a company has provided the practical teaching on business values, processes, restrictions and goals, in which has been proven a great opportunity to grow professionally. I

have also been given the chance to take part in the company workshops, seminars and events, which provide both technical and business contents and have contributed interesting, useful, detailed insights on technology and business strategies and concepts.

Being English the most often spoken language in itestra, my language skills are also better at the end of this period.

To sum up, I consider the proposed goals achieved, as well as set as a basis for the accomplishment of further objectives.

## 5.2 Application of learnt concepts and knowledge

Teachers in the first year often insisted on how it is not the goal of a university degree to make us learn absolutely every single concept we might need in a future but to teach us "to think like engineers". Thus, while learning how to operate on signals, calculate wave reflexion or program infinite loops we acquired more important competencies: we learnt to properly specify problems, extract the important data for their processing and find the necessary information in order to solve them.

This ability to analyse engineering problems is, in my opinion, among the most important competencies I have acquired –still to be improved–, as it covers a wide area of knowledges and is the base to solve any known or unknown problem.

I also appraise an argumentation that, despite seeming trivial, took me almost three years to learn: something being difficult does not mean that only the geniuses can achieve it, it only means that you will have to work harder to get it. As obvious as it sounds, my experience says that we tend not to believe it, becoming this thought one of the hardest obstacles in the achievement of our goals. During the last half of my degree, I have verified this argument, and specially in the beginning of the project, which is always the hardest part, focusing on work instead of difficulty has become a great momentum.

With respect to technical knowledge, two areas can be distinguished:

- The first one concerns technologies and programming skills. For it, concepts learnt in this subjects have been relevant:

– *Fundamentos de los Computadores* (3rd year) and *Sistemas Operativos* (4th and 5th

- year) dealt with basis of hardware and computing.
- *Fundamentos de la Programación* (1st year) and *Metodología de la Programación* (3rd year) provided teaching on programming, its basis, logic and algorithms.
  - *Sistemas Telemáticos* (2nd and 3rd year), *Información Audiovisual en Redes de Ordenadores* (4th year) and *Planificación y Gestión de Redes de Ordenadores* (6th year) address issues related to network administration. Its lessons provided detailed insights on state-of-the-art technologies and algorithms, widening our perspective of the current landscape and improving our capabilities on design and implementation.
  - Diverse subjects on electronics, data transmission and processing, and calculus provided complementary competencies for the analysis and comprehension of the whole picture regarding logic, analysis and understanding of limits and possibilities of the project.
- The second one regards project management and business logic, and it has been influenced by three focuses:
    - *Proyectos* (6th year), approaching topics directly related to project management: basis, history, modalities, analysis, implementation...
    - *Licenciatura en Administración y Dirección de Empresas*, covering all the business aspects.
    - Different projects and assignments carried out in most of the subjects of the degree, providing practical teaching on project management, teamwork, etc.

### 5.3 Acquired competencies

This section can be conceived as the sum up of the previous two: the accomplishment of the proposed goals has been accompanied by the acquirement of the underlying capabilities and knowledge. Therefore, as a summary, the project has provided the opportunity to improve my technical competencies by researching on the goals, taking decisions on design and implementing the mentioned tool, in an environment that has made a business oriented vision of the whole process possible, an aspect that I consider an essential contribution to the knowledges of my complementary degree on Business.

It is fair to say that this experience has contributed to my growth as an engineer, a professional and as a person, even if it is just the very beginning.

# Acronyms

**ALM** Application Lifecycle Management

**API** Application Programming Interface

**DB** Database

**CSV** Comma-Separated Values

**IDE** Integrated Development Environment

**NASCIO** National Association of State Chief Information Officers

**IO** Input/Output

**I/O** Input/Output

**IT** Information Technology

**MD5** Message-Digest Algorithm 5

**MIME** Multipurpose Internet Mail Extensions

**PDE** Plug-in Development Environment

**RAP** Remote Application Platform

**RCP** Rich Client Platform

**RFC** Request For Comments

**UI** User Interface

**URL** Uniform Resource Locator





# Bibliography

[1] NASCIO 2008 Survey of the States. *Digital States at risk! Modernizing Legacy Systems.*

<http://www.nascio.org/publications/documents/NASCIO-DigitalStatesAtRisk.pdf>

[2] Lamb, J. (2008). "Legacy systems continue to have a place in the enterprise." *Computer Weekly.*

<http://www.computerweekly.com/feature/Legacy-systems-continue-to-have-a-place-in-the-enterprise>

[3] itestra GmbH

<http://www.itestra.de>

[4] RFC-4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files

<http://tools.ietf.org/html/rfc4180>

[5] Apache Software Foundation

<http://www.apache.org>

[6] Apache Commons

<http://commons.apache.org>

[7] Apache Logging

<http://logging.apache.org>

[8] StackOverflow

<http://www.stackoverflow.com>

[9] StackExchange

<http://www.stackexchange.com>

- [10] Salvatierra, G., Mateos, C., Crasso, M, Zunino, A. and Campo, M. (2013). "Legacy System Migration Approaches" *IEEE Latin America transactions*, Vol II, Nr. 2.  
[http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol11/vol11issue2March2013/11TLA2\\_22Salvatierra.pdf](http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol11/vol11issue2March2013/11TLA2_22Salvatierra.pdf)
- [11] Pronschinske, M. (2009) "Top ALM Tools and Solution providers" *DZone*  
[architects.dzone.com/news/top-10-alm-solutions](http://architects.dzone.com/news/top-10-alm-solutions)
- [12] Toll, W. (2014) "Top 48 Integrated Developer Environments (IDEs) & Core Editors"  
*textitProfitBricks*  
<http://blog.profitbricks.com/top-integrated-developer-environments-ides/>
- [13] "Platform Plug-in developer guide" *Eclipse documentation*  
[http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fguide%2Ftools%2Fproject\\_wizards%2Fplugin\\_structure.htm](http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.pde.doc.user%2Fguide%2Ftools%2Fproject_wizards%2Fplugin_structure.htm)
- [14] Burnette, E. *Eclipse IDE Pocket Guide*. O'Reilly, 2005.
- [15] Holzner, S. *Eclipse*. O'Reilly, 2004.
- [16] Eclipse Foundation  
<https://eclipse.org/pde>