HAL
open science

CHOReOS Integration Plan - 1st version (D5.7.1)

Amira Ben Hamida, Jean-Pierre Lorré

To cite this version:

Amira Ben Hamida, Jean-Pierre Lorré. CHOReOS Integration Plan - 1st version (D5.7.1). 2011. hal-00664309

HAL Id: hal-00664309
https://inria.hal.science/hal-00664309v1

Preprint submitted on 30 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CHOREOS
Large Scale Choreographies for the Future Internet

ICT IP Project

Deliverable D5.7.1

Integration Plan – 1st version

http://www.choreos.eu

template v8

Project Number	:	FP7-257178
Project Title	:	CHOReOS Large Scale Choreographies for the Future Internet

Deliverable Number	:	D5.7.1
Title of Deliverable	:	Integration Plan – 1st version
Nature of Deliverable	:	Report
Dissemination level	:	Public
License	:	Creative Commons Attribution 3.0 License
Version	:	A
Contractual Delivery Date	:	1st October 2011
Contributing WP	:	WP5
Editor(s)	:	Jean Pierre Lorré (EBM)
Author(s)	:	Amira Ben Hamida (EBM), Jean Pierre Lorré (EBM)
Reviewer(s)	:	Marco Aurélio Gerosa (USP), Valérie Issarny (INRIA)

Abstract

The Integration Plan details the process that will be followed to release the CHOReOS Integrated Development and Runtime Environment (IDRE) software. As a first step, the Specification of the CHOReOS IDRE is described in D5.2. As a second step, we provide the integration plan detailing the steps and significant milestones. Then, we provide development guidelines. Finally, we survey the collaborative development platforms that facilitate the integration activities.

Keyword list

Integration Plan, Milestones, Components, IDRE, Collaborative Development, Assessment.

Document History

Version	Changes	Author(s)
1.0	Document creation	Jean-Pierre Lorré Amira Ben Hamida
2.0	Version for review	Jean-Pierre Lorré Amira Ben Hamida
2.1	Version after review feedbacks	Jean-Pierre Lorré Amira Ben Hamida
3.0	Final version for PTC	Jean-Pierre Lorré

Document Review

Review	Date	Ver.	Reviewers	Comments
Outline	03/10/11	1.0	J.P. LORRE (EBM)	
Draft	05/10/11	2.0	J.P. LORRE (EBM)	
QA	19/10/11	3.0	M.A. GEROSA (USP), H. VINCENT (THA), V. ISSARNY (INRIA), J.P. LORRE (EBM)	
PTC	20/10/11	A	PTC	

Glossary, acronyms & abbreviations

Item	Description
CA	Consortium Agreement
DL	Deliverable Leader
DOW	Description of Work
IAC	Industrial Advisory Committee
MST	Management Support Team
OSS	Open Source Software
PL	Project Leader
PMC	Project Management Committee
PO	Project Officer
PTC	Project Technical Committee
SL	Scientific Leader
VCS	Version Control System
WP	Work Package
WPL	Work Package Leader

Table of Contents

1. Introduction 1

1.1. What is an Integration Plan? 1
1.2. Reading Key 1

2. CHOReOS IDRE Specification 2

2.1. Overview 2
2.2. CHOReOS Components 3

3. Integration Schedule 4

3.1. DoW Deadlines 4
3.2. Gantt Diagram for Integration 4
3.3. Test Bed and Assessments 6

4. Development Guidelines 7

4.1. Programming language: Java 7
4.2. Logging tooling: Log4j 7
4.3. Building tooling: Maven 8
4.4. Integrated Development Environment: Eclipse 9
4.4.1. Maven integration 9
4.4.2. Subversion integration 9

5. Collaboration tools 10

5.1. Revision control system: Apache Subversion 10
5.1.1. Directory hierarchy 11
5.1.2. Branching policy 11
5.2. Issue Reporting and Tracking: Jira 11
5.3. Continuous Integration: Bamboo 13
5.4. Collaborative Development Environments: the OW2 forge 13

6. Conclusion 15

1. Introduction

1.1. What is an Integration Plan?

The purpose of the Integration Plan is to define the order in which the components and subsystems of the CHOReOS IDRE should be integrated, which builds to create when integrating the system, and how they are to be assessed.

The main stakeholders involved are the WP5 leader and co-leader, the developers, and the test designers.

1.2. Reading Key

This document is organised as follow:

- The first part provides a brief overview of the CHOReOS IDRE software architecture.
- The second part deals with the integration schedule.
- The third part describes the integration steps as well as the assessment strategy.
- The fourth part provides information about the CHOReOS collaborative development platform.
- The last part concludes the document.

2. CHOReOS IDRE Specification

This section provides information about the CHOReOS IDRE architecture, which is detailed in D5.2 “Specification of the CHOReOS IDRE”.

2.1. Overview

According to the deliverable D5.2, Figure 1 depicts CHOReOS “big picture” that contains the main CHOReOS software components as listed in the following table.

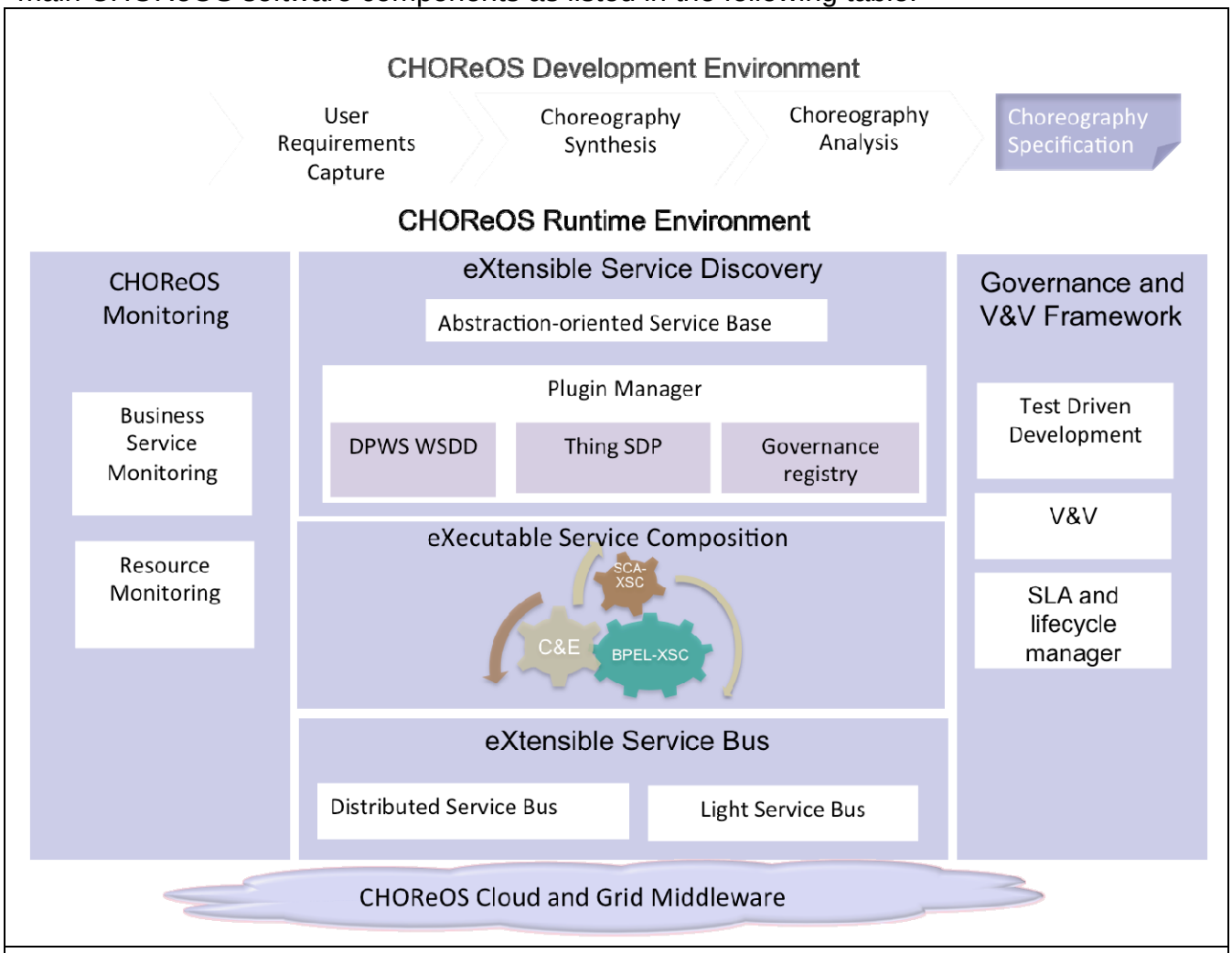

Figure 1: CHOReOS IDRE Overview

2.2. CHOReOS Components

This table gives the exhaustive list of the CHOReOS components:

CHOReOS Subsystem	Components	Deliverables
ULS Choreography Development	Requirements Specification Tools	D2.2, D2.3
	Synthesis Processor	
	Choreography Analyser	
Extensible Service Discovery	Abstraction Oriented Service Base Management	D3.2.1, D3.2.2, D3.3, D4.3
	Plugin Manager	
	Governance Registry for Business Services	
	Things Discovery Protocol	
Governance Framework and V&V	SLA and Lifecycle Manager	D4.3, D4.2.1, D4.2.2, D4.3
	V&V Components	
	TDD Framework	
	CHOReOS Monitoring	
Extensible Service Composition	BPEL-XSC	D3.2.1, D3.2.2
	Composition and Estimation	
	SCA-XSC	
	Reconfiguration Management for Service Substitution	
Extensible Service Access	Distributed Service Bus	D3.2.1, D3.2.2
	Light Service Bus	
	DSB-LSB Bridge	
	Enactment Engine	
CHOReOS Monitoring	Business Service Monitoring	D4.2.1, D4.2.2, D4.3
	Resource Monitoring	
	Complex Event Processor	

Table 1: CHOReOS software components

3. Integration Schedule

The aim of this section is to present the steps involved in the integration of the components coming from the technical work-packages (WPs 2, 3, 4) and to provide the corresponding platform to the uses-cases (WPs 6, 7, 8) for assessment.

3.1. DoW Deadlines

According to the DOW the following milestones has been defined:

WP2 Dynamic development of adaptable, QoS-aware ULS choreographies			
D2.2	Definition of the dynamic development process for adaptable QoS-aware ULS choreographies	UDA	M24
D2.3	CHOReOS dynamic development process: methods and tools	UDA	M36
WP3 Service-Oriented Middleware for the Future Internet			
D3.2.1	CHOReOS middleware first implementation	EBM	M18
D3.2.2	CHOReOS middleware implementation	EBM	M24
D3.3	Integrated CHOReOS middleware and deployment of ULS, QoS-aware adaptive choreographies	USP	M36
WP4 Governance and V&V support for choreographies for the Future Internet			
D4.2.1	V&V tools and infrastructure – strategies, architecture and first implementation	CNR-ISTI	M18
D4.2.2	V&V tools and infrastructure – strategies, architecture and implementation	CNR-ISTI	M24
D4.3	Final release of the V&V tools and infrastructure	CNR-ISTI	M30
WP5 CHOReOS IDRE - Integrated Development and Runtime Environment			
D5.3.1	CHOReOS IDRE and user manual – 1st version	EBM	M24
D5.4	Implementation of the test-bed	EBM	M24
D5.3.2	CHOReOS IDRE and user manual – revised version	EBM	M30
D5.5	CHOReOS IDRE as open-source packages	EBM	M30
D5.6	Final version and assessment of the CHOReOS IDRE	MLS	M36

Table 2: CHOReOS milestones

3.2. Gantt Diagram for Integration

The following Gantt diagram depicts the CHOReOS integration process.

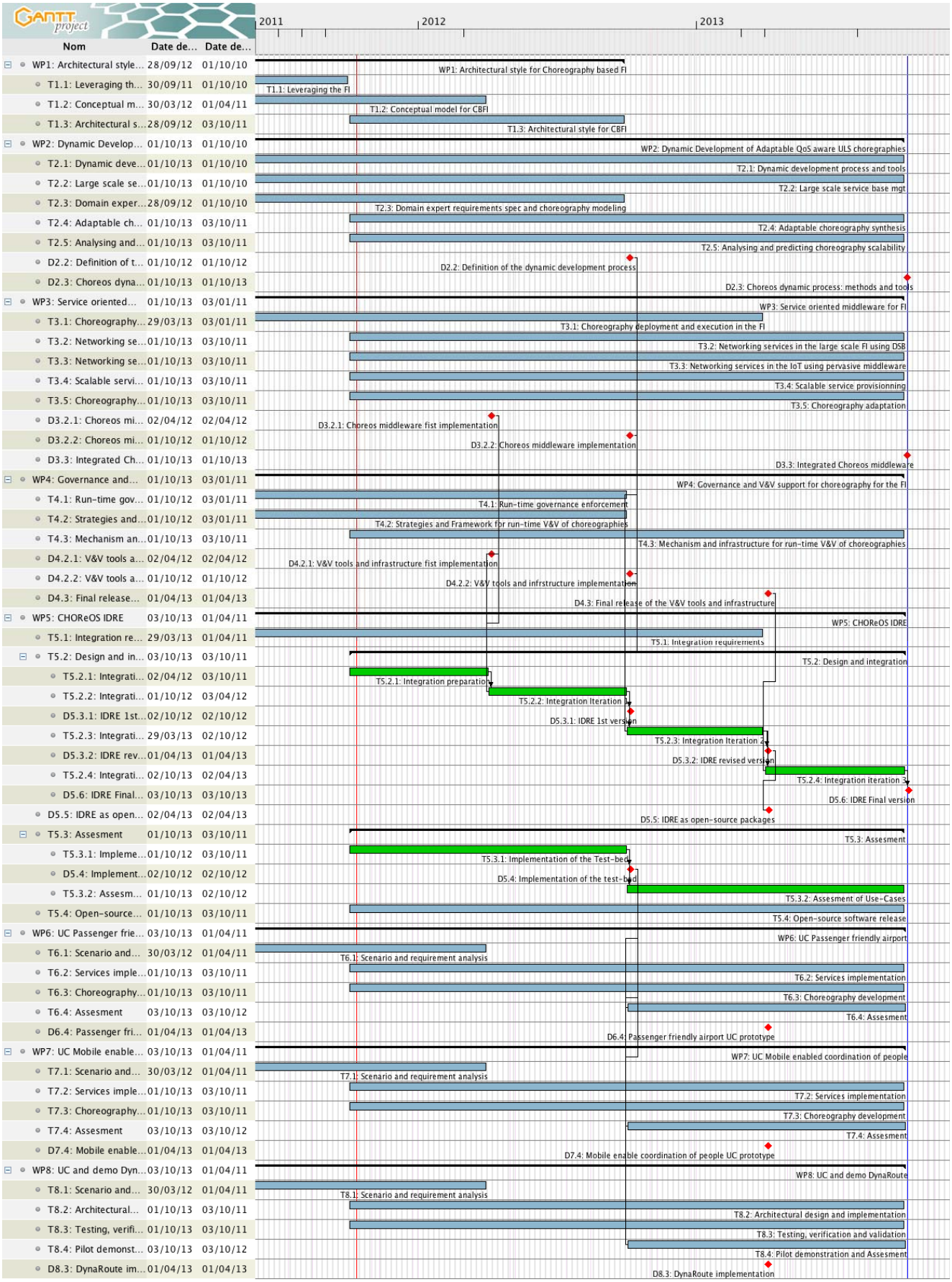

Figure 2: CHOReOS Gantt diagram

cc BY

The integration work takes place along the following tasks:

- Task 5.1: Integration requirements; this task is mainly about collecting requirements that will be use during integration.
- Task 5.2: Design and integration; which is about the main IDRE integration work.
- Task 5.3: Assessment; that prepares and manage the overall assessment of the IDRE.
- Task 5.4: Open-Source software releases; this task takes in charge the software packaging of the IDRE in order to produce open-source releases.

The design and integration task (T5.2) has been split into four sub tasks in order to smoothly integrate CHOReOS software components as far as they become available:

- T5.2.1 [M12:M18]: Integration preparation: this sub-task aims to prepare the supporting environment for the integration.
- T5.2.2 [M18:M24]: Integration iteration 1: this sub-task aims to produce the first CHOReOS integrated platform taking into account components coming from WP3 (D3.2.1) and WP4 (D4.2.1).
  Although WP2 will not have produced so far an integrated set of components to include in this first release, the coordination delegate specification will be released in order to enable separate work between WP2 and other work-packages.
- T5.2.3 [M24:M30]: Integration iteration 2: this sub-task aims to produce the first complete CHOReOS integrated platform based on components from WP2, WP3 and WP4. This software platform will be released as an open-source package.
- T5.2.4 [M30:M36]: Integration iteration 3: this sub-task aims to produce the final CHOReOS integrated platform based on updated components that take into account feedback coming from the Use-Case assessment tasks.

3.3. Test Bed and Assessments

The assessment task (T5.3) aims to evaluate the execution of the CHOReOS runtime environment, thanks to choreographies design in Use-Cases. It is organized according to two sub-tasks:

- T5.3.1 [M12:M24]: Implementation of the test bed: this sub-task aims to develop probes, interceptors and service mock-ups that are necessary for the assessment of the Use-Cases' choreographies.
- T5.3.2 [M24:M36]: Assessment of Use-Cases: this sub-task aims to support the technical assessment carried out by Use-Cases' work-packages (tasks T6.4, T7.4 and T8.4).

4. Development Guidelines

The use of a plethora of programming languages, combined with various OS-level integration scripts, increases the complexity of the deployment and maintenance of software. The CHOReOS project will avoid this pitfall by using small subsets of the possible coding languages and using others if and only if they offer significant improvements or are required by some other external factor. In this section, we present specific languages of choice for CHOReOS and prescribe tools and good practices that we expect the developers to adhere to when contributing to the code.

We however do not constrain any programmer to this set of tools. An individual is welcome to use any of the tools, as long as s/he does not expect any support and does not cause any stalls in the development. In short, we expect a unified level of productivity from each of the contributors to the project codebase and full responsibility of the programmers.

The only exceptions to the aforementioned rules are the programming languages and coding styles as well as maven for builds management. The use of additional languages will have to be strongly argued and ultimately authorised by PTC membership.

The adherence to the coding style is of utmost importance. Badly written and undocumented code will not be tolerated since it makes code maintenance very hard. Unified coding style will greatly benefit the current programmers on the project and also increase chances of future adoption within the F/OSS community

4.1. Programming language: Java

By convention, Java source projects are organized in the following way:

- all the source code (i.e. the .java files) belong into the src/ directory;
- the resources, such as the images, the international strings etc., are contained in the res/ directory; and
- the compilation results are written into the bin/ or target/ directory.

Rather than creating the directories manually, we recommend using Maven (see next subsection) that, on top of other features, can create the full project skeleton. Another reason for using Maven is its support for creating projects for popular IDEs.

Best coding practices are a wide topic, often also subject to debate. We therefore encourage that all developers refer to Coding Best Practices[1] by R.Rajesh Kannan.

4.2. Logging tooling: Log4j

Log4j[2] offers a hierarchical way to insert logging statements within a Java program. Multiple output formats and multiple levels of logging information are available. By using a dedicated logging package, the overhead of maintaining thousands of System.out.println statements is alleviated as the logging may be controlled at runtime from configuration scripts.

[1] http://www.scribd.com/doc/8526130/
[2] http://en.wikipedia.org/wiki/Log4j

4.3. Building tooling: Maven

The programming languages come with basic libraries and tools for compiling the code and running the software. The developers are by default free to organize the code in any way they know or like, but when the complexity of a software project raise, the manageability of its code becomes questionable. To help with this problem, a set of tools that supplement those belonging to the language itself provide the possibility to formalize the organization of the sources, turn common tasks such as compiling and testing code into a compact configuration, and effectively make the use of code much more portable. In this section we focus on Maven from the Apache foundation.

Maven[3] is a tool that handles project management and builds automation. It is mostly used for Java development, but can also handle projects in C#, Ruby, Scala, and other languages. It has a similar purpose as the Apache Ant[4] tool.

Maven is based on the concept of Project Object Model (POM). The POM, which is an XML file, describes the software project, the dependencies from other external modules, as well as the build order. Maven has predefined targets allowing to perform specific tasks (e.g., code compilation, code packaging etc.).

The next figure shows a very simple example of a POM file (by default named pom.xml). This is sufficient to build the project and to run the associated unit tests. Each <dependency> block references an artefact (usually a .jar library) that is stored on a public repository. When a library or a plugin is needed by a project, a query should be submitted to a search engine (see below) copying and pasting the corresponding <dependency> block into the project pom.xml. A list of the most common Maven plugins is available at Apache website[5]. More information and detailed documentation is available at the Maven complete reference website[6].

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

Figure 3: Simple example of Maven pom.xml

[3] http://en.wikipedia.org/wiki/Apache_Maven
[4] http://en.wikipedia.org/wiki/Apache_Ant
[5] http://maven.apache.org/plugins/index.html
[6] http://www.sonatype.com/books/mvnref-book/reference/public-book.html

In CHOReOS, the POM org.ow2.choreos.choreos contains all the default settings for the project, and must be inherited by every module using Maven as a build tool.

When looking for available artefacts, we recommend using these search engines:

- http://repository.ow2.org/nexus/index.html
- http://mvnrepository.com/

4.4. Integrated Development Environment: Eclipse

Integrated development environments (IDEs) comprise tools or software suites that integrate various tools, including the editor, compiler, debugger and other development essentials. In this section we describe Eclipse, one of the most commonly used IDEs.

Eclipse is a software development environment that supports multiple programming languages and combines an integrated development environment (IDE) with an extensible plug-in system. Although it's written mostly in Java, one can use Eclipse to develop in any language. As opposed to applications where most functionality is hard-coded, Eclipse uses plug-ins for most of its functionalities in addition to the basic runtime system. Eclipse's runtime system is based on an OSGi[7] compliant implementation called Equinox. OSGi Alliance is a non-profit corporation that provides open specifications, reference implementations and test suites for modular assembly of software built with Java technology.

More information and detailed documentation is available at the Eclipse Marketplace website[8].

4.4.1. Maven integration

m2eclipse[9] provides comprehensive Maven integration for Eclipse. Developer can use m2eclipse to manage both simple and multi-module Maven projects, execute Maven builds via the Eclipse interface, and interact with Maven repositories. m2eclipse makes development easier by integrating data from a project's Object Model with Eclipse IDE features.

4.4.2. Subversion integration

Subclipse[10] integrates Subversion into Eclipse as an Eclipse Team Provider plugin. It facilitates the use of SVN directly in Eclipse IDE with its own graphical user interface.

[7] http://www.osgi.org/Main/HomePage.
[8] http://marketplace.eclipse.org/
[9] http://m2eclipse.sonatype.org/
[10] http://subclipse.tigris.org/

5. Collaboration tools

In the software development cycle, the sooner the code enters a common development pool, the more other developers use this code, be it as a library imported into another component, or as a service communicating from another process or node. The collaboration tools provide essential support for keeping and tracking the changes in the code, automatically detecting possible problems in the code, planning the releases and reporting bugs.

In this section we first provide the description of the repositories that support the storage, versioning and exchange of the code. For the development coordination, we present feature request and problem reporting tools, useful also for the external testers and users for reporting any issues found in the software. Finally, to reduce the chance of issues discovered later, we describe additional tools that support continuous integration.

5.1. Revision control system: Apache Subversion

Revision control software takes care of changes in documents, programs and other information stored as files on the file system. This way, developers do not have to manually control versions when saving newer versions of the files on the server. Distributed revision control (Git, Mercurial) takes a peer-to-peer approach, as opposed to the client-server approach of centralized systems (SVN, CVS).

In CHOReOS we decided to use a centralized VCS, namely SVN, described in more detail below. This means that SVN is the only official and supported repository, and the developers must commit their code regularly. The reasons for this decision is that we believe that the distributed revision control is more complex than the centralized one, thus we expect that the developers will have less difficulties using SVN than they would have using Git or Mercurial. Further, we want to encourage one central location of code that is committed often by all contributors.

Apache Subversion[11] (also known as SVN) was founded in 2000 by CollabNet Inc. It is a software versioning and a revision control system used to manage revisions of source code and other documents (text and binary files). It is mostly-compatible successor to the widely used Concurrent Versions System[12] (CVS). Subversion uses the Apache License, making it free software and open source.

In the CHOReOS project, it has been agreed that SVN will be the version control system used in the central forge, however developers are free to use Git or Mercurial locally.

A cheat sheet of SVN commands is available online at the Abbey Workshop website[13]. More information and detailed documentation is available at the Subversion book website[14].

[11] http://en.wikipedia.org/wiki/Apache_Subversion
[12] http://en.wikipedia.org/wiki/Concurrent_Versions_System
[13] http://www.abbeyworkshop.com/howto/misc/svn01/
[14] http://svnbook.red-bean.com/

5.1.1. Directory hierarchy

Because in Subversion regular directory copies are used for branching and tagging, we decided in CHOReOS to choose a repository location for each subproject root (i.e., the ”topmost“ directory that contains data related to that subproject) and then create three subdirectories beneath that root:

- **trunk** meaning the directory under which the main project development occurs;
- **branches** which is a directory in which various named branches of the main development line may be created;
- **tags** which is a collection of tree snapshots that are created, and perhaps destroyed, but never changed.

5.1.2. Branching policy

In the CHOReOS project, it is strongly recommended that active development happen in the trunk. Changes made to trunk have the highest visibility and get the greatest amount of exercise that can be expected from unreleased code. For this to be beneficial to everyone, the trunk is expected at all times to be stable. It MUST build. It MUST work. It might not be release-ready, but it MUST certainly be test-suite ready.

We also strongly recommend seeing large changes broken up into several, smaller, logical commits — each of which is expected to meet the aforementioned requirements of stability.

Yet, we understand that it can be nearly impossible to apply all these policies to particularly large changes (new features, sweeping code reorganizations, etc.). It is in those situations that developers might consider using a custom branch dedicated to their development task.

5.2. Issue Reporting and Tracking: Jira

The tools for issue reporting and tracking act as a smart pinboard that can be used by the developers, release managers and the users of the software. They are therefore highly important in directing the development of a large software project as a place to record the progress of the development, publish the feature requests, track the issues and their status, and direct the integration of components.

In CHOReOS, we will use Atlassian JIRA, as it comes with the OW2 community.

JIRA[15] is a tool used for bug tracking, issue tracking, and project management. It is a proprietary product, developed by Atlassian. Atlassian provides JIRA for free to open source projects.

The JIRA interface provides the following bug-tracking features:

- View issue details including custom fields, attachments, workflow actions and recent activity.
- Create new bugs from browser, email, IDE or smart phone client.
- Quickly triage issues with auto-complete entry fields for labels, components, and versions.
- Leverage activity history to quickly access recently opened issues, projects and searches.

[15] http://en.wikipedia.org/wiki/JIRA

Atlassian provides IDE Connectors with which the developers can interact with JIRA from Eclipse[16]. Third-party integrations are available for other IDEs, for example, NetBeans[17]. This enables the exchange of information between the code repository and the issue reports, making the development and bug fixing progress easier and more coordinated.

Projects metadata

JIRA projects organize bugs into components for sub-grouping, and into versions for identifying affected and target fix releases.

From any project, it is possible to access:

- Issue statistics and key metrics for monitoring progress.
- Activity streams displaying recent changes across bugs, issues, projects or people.
- Charts and reports showing the most popular issues, recently opened/closed issues and more.
- Road Map highlighting issues remaining in current or future versions.
- Change Log detailing bugs and issues fixed in previous versions.

It is also possible to set up a wallboard for each team or to use JIRA Query Language (or JQL) to create reports and charts and customize dashboards. Events can be subscribed to via RSS or email notifications.

Figure shows a screenshot of the Atlassian web interface. More information and detailed documentation is available at the Atlassian JIRA website[18].

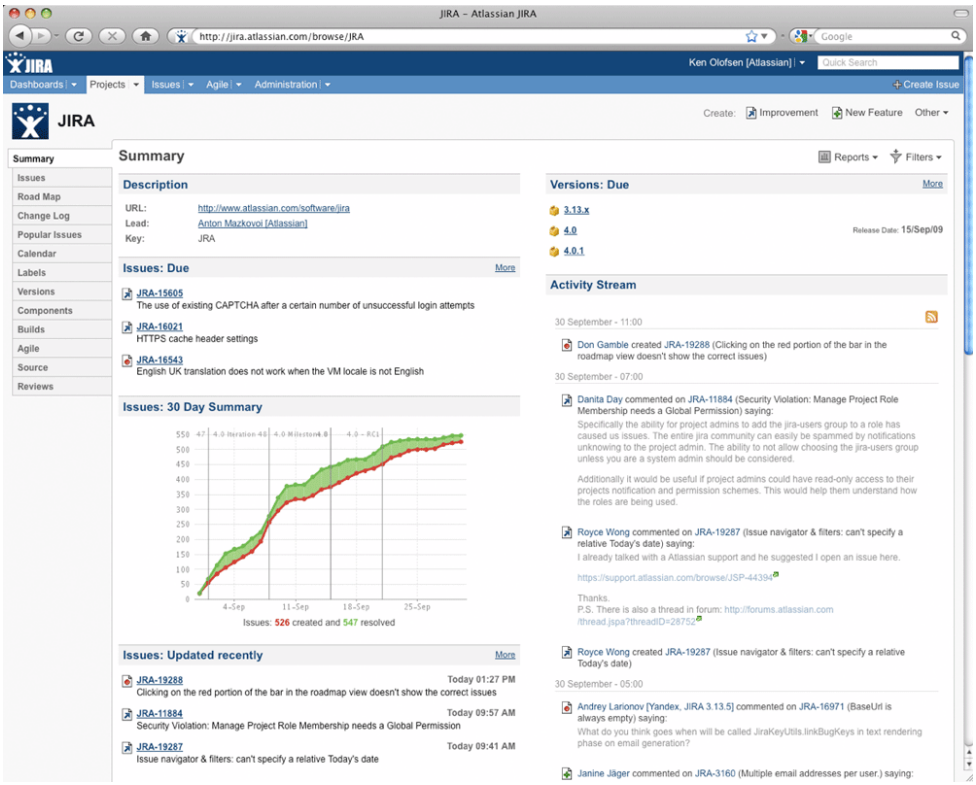

Figure 4: Screenshot of Jira Web Interface

[16] http://en.wikipedia.org/wiki/Eclipse_(software)
[17] http://netbeans.org/
[18] http://www.atlassian.com/software/jira/

5.3. Continuous Integration: Bamboo

Continuous Integration[19] (CI) comprises practices such as daily builds and additional checks to prevent bugs. In order to enable automatic daily builds, Continuous Integration software gather the whole source in one place (with different revisions), automate the build process and testing, and provide the latest working executable to anyone involved in the project. The CI model comprises a set of activities for the process implementation: building the system, running tests, deployment activities, and finally reporting test and deployment results. Of course, CI does not guarantee bug-free software, it only provides a way to reduce the time to eliminate as many bugs in the code as possible.

There are many known CI software products and in CHOReOS we will use Bamboo since it is the one provided by the OW2 platform.

Bamboo[20] is a continuous integration server from Atlassian and is free for philanthropic and open-source projects.

Bamboo is tied to no specific programming language or build tool. The range it supports is very broad and includes ant, maven, make, and any command line tools. It provides build notifications that can be customized by type of event. Notifications are sent via email, RSS, instant message, or shown in pop-up windows in Eclipse-based IDEs.

Bamboo can be installed as a standalone server, which is prepackaged with Jetty application server and removes all difficulties related to configuration.

5.4. Collaborative Development Environments: the OW2 forge

The collaboration tools described so far need to be hosted in some central location. It is therefore best to use an already existing site that integrates many or all of the needed tools. Such sites are called forges, and we will use the OW2 forge to host all the CHOReOS code and public project-related content:

- The code of the components developed during the CHOReOS project.
- The scripts and specifications required for building the packages for the target distributions (e.g. deb for Debian Package or rpm for Red Package Manager).
- Documentation, providing the guides and manual for the end-users and administrations.
- The structure of the repository for the component code will follow from the architecture agreed in the relevant WP. In this way, the closely coupled components will reside within the relevant module. Each module will contain the subfolders standard for the Subversion modules:
  - trunk: the folder containing the current development version of the module.
  - branches: this folder contains an arbitrary number of subfolders, each of which is a copy of the contents of the trunk at the time of the creation. These subfolders can represent the experimental development that needs to be tested and validated

[19] http://en.wikipedia.org/wiki/Continuous_integration
[20] http://www.atlassian.com/software/bamboo/

before it can be merged back into trunk. It usually also contains the maintenance of the past releases of the module.

- tags: have the subfolders that are similar in content as those in the branches, but should only contain snapshots of the trunk or a branch with no purpose of being changed. The tags usually contain the state of the code at the points of versions or releases.

6. Conclusion

This document is the first integration plan for CHOReOS, delivered at the end of the first year of the project. It provides main steps that will be carried out by partners in order to integrate the CHOReOS software platform. It provides information about the collaborative development platform that will be provided by the OW2 consortium. The resulting integrated software platform will be exploited as an open-source project.

An update of this document will be provided at M24 (the end of the second year of the project). It will adjust the current plan according to the available results.