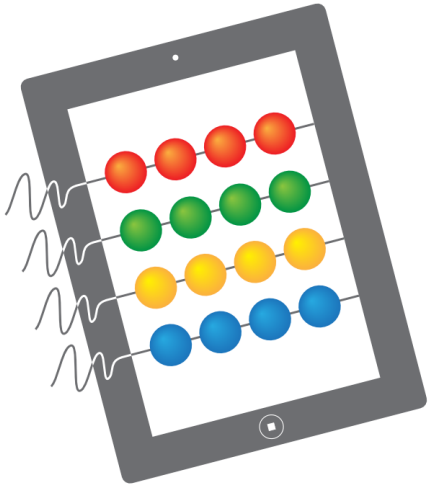
FP7 ICT STREP Project



LEARN PAD

Deliverable D7.1

Project best practices, support tools and integration plan

http://www.learnpad.eu

LinAgora · BeC · No Magic Europe · Fachhochschule Nordwestschweiz · UNICAM · WIKI

LaTeX template v. 0.5

Project Number	:	FP7-619583
Project Title	:	Learn PAd Model-Based Social Learning for Public Administrations

Deliverable Number	:	D7.1
Title of Deliverable	:	Project best practices, support tools and integration plan
Nature of Deliverable	:	Report
Dissemination level	:	Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	4.0
Contractual Delivery Date	:	31 July 2014
Actual Delivery Date	:	31 July 2014
Contributing WP	:	WP7
Editor(s)	:	Robert Woitsch (BOC)
Author(s)	:	Vedran Hrgovcic (BOC), Robert Woitsch (BOC), Fabio Mancinelli (XWIKI), Darius Silingas (NME), Jean Simard (XWIKI), Caleb James Delisle (XWIKI)
Reviewer(s)	:	Jean-Pierre Lorré (LIN), Guglielmo De Angelis (CNR)

Abstract

This deliverable introduces recommendations and guidelines on how Learn PAd partners collaborate, develop, integrate and test their software in an efficient, flexible and reliable way. It targets the exchange of best practices in terms of software development and integration in the context of such a heterogeneous platform like Learn PAd and taking into account the different expertise of the partners.

Keyword List

Software development, Tools, Practices, Integration Plan

Document History

Version	Changes	Author(s)
0.1	First ToC	Vedran Hrgovcic, Fabio Mancinelli
0.2	Switch to the LaTeXTemplate	Jean Simard
1.0	First draft	Jean Simard, Caleb James Delisle, Fabio Mancinelli
1.1	Refined ToC	Robert Woitsch
2.0	Internal review version	All authors
4.0	Internal review comments taken into account	Fabio Mancinelli

Document Reviews

Release	Date	Ver.	Reviewers	Comments
ToC	5 Jun 2014	0.1	N.A.	N.A.
Draft	20 Jun 2014	1.1	Robert Woitsch	Many missing parts, ToC must be refined
Internal	22 Jul 2014	2.0	Jean-Pierre Lorré, Guglielmo De Angelis	Comments in the review reports available from the internal wiki
Candidate Final	29 Jul 2014	4.0	Guglielmo De Angelis	

Table Of Contents

List Of Figures

1 Introduction

This deliverable introduces recommendations and guidelines on how Learn PAd partners collaborate, develop, integrate and test their software in an efficient, flexible and reliable way. It targets the exchange of best practices in terms of software development and integration in the context of such a heterogeneous platform like Learn PAd and taking into account the different expertise of the partners. It hence has to clearly separate different software development approaches like the one used in open source software development, the provision of closed source platforms including extensions as well as add-in implementation that is expected to be necessary for part of legacy integration. The Learn PAd platform therefore does not only deal with novel software development but also with different development approaches dealing with different development timelines and different accessibility of software with the aim to continuously provide an integrated, tested and hence demonstrated able Learn PAd platform. Therefore, this document defines high level rules that provide each development team the maximum of required freedom to follow each individual development approach, but defines how the different development teams work together by introducing an overview of the Learn PAd system, introducing the relevant development processes and presenting the provided development infrastructure. Continuous integration is discussed in the context of the work plan and will be further elaborated when the architecture that is currently developed in WP2 is more mature.

1.1. Structure of the deliverable

The document is organized as follows:

- Section 2 justifies the choice of a SOA architecture for the Learn PAd platform. Also it discusses the general choices related to both Open Source, and Closed Source development.
- Section 3 remarks what are the high level building blocks identified from the requirement analysis done in Deliverable D1.1 [3]. The goal is to define the context addressed by this deliverable.
- Section 4 focuses on the processes that are proposed for managing the development of software deliverables from their design to the publication of a stable release.
- Section 5 describes the infrastructure elements that the consortium will use in order to implement the previouslly defined processes.
- Section 6 proposes an integration plan the consortium will abide by in order to schedule the releases and manage the evolution of the software deliverables.
- Section 7 draws both a summary, and some conclusions.

2 Preamble

Learning and Knowledge Management systems are typical socio - technical systems, as the learning process and the knowledge evolution happen mainly in the head of humans. Hence, although the focus of this deliverable is the technical collaboration between partners, socio-technical aspects need as well to be considered, as described in Deliverable D1.1 [3], in order to correctly reflect the ecosystem around the platform. In general the Learn PAd project distinguishes:

- **The Learn PAd system** describes the socio-technical ecosystem and hence includes the technical components, the content, the organizational context and processes as well as the conceptual thinking paradigm. It is necessary to consider the importance of evolving meta models that lead to evolving data schema, as well as evolving models that lead to evolving system configurations. Hence both aspect that are not directly related to technology but have a huge impact on the software.
- **The Learn PAd platform** describes all technical aspects starting with the core building blocks of the Learn PAd platform, but also includes the interaction with legacy systems. It is necessary to separate the major building blocks, as those building blocks are individual sub-systems concerning with different aspects, using different technology and platforms, using different development approaches and are developed by different development teams with different educations and mindsets.
- **The Learn PAd concept** describes the bridge between the Learn PAd system and the Learn PAd platform, by introducing conceptual and semantic modelling. Those models influence the platform in form of different data schema and different system configuration.
- **The Learn PAd usage** describes how the Learn PAd system can be introduced and applied into an organization and considers cultural, organizational, change management and legal aspects when introducing such a system. With respect to software development, the Learn PAd usage is the same as any other complex software system.

2.1. Service Oriented Architecture

As it will be described in Section 3 the Learn PAd platform will be based on different building blocks.

Although the specification of the architecture will be performed in WP2, the high level separation of concern clearly identifies that each of the building blocks is of a completely different nature.

Hence Learn PAd follows the Service Oriented Architecture by distinguishing between four independent and autonomous software systems that guarantee an independent execution and commonly agreed interfaces for data exchange. Hence we identify interfaces between the modelling and the core environment for model data reading as well as between the core and the assessment environment for log data transfer. Those interfaces are guaranteed by proxy services using Web-Services. The interaction with those proxy services is however matter of individual implementation.

The Service Oriented Approach has not only been selected to enable independent development and execution of the main building blocks, but also to enable the exchange of different implemented

solutions within those building blocks, providing flexibility of the Learn PAd platform, by a customer-specific selection of tools.

2.2. Software Development Projects

In order to reflect the different nature of major building blocks of the Learn PAd project, it will consist of two major types of software development projects, each having their own development strategies and technologies.

- Core Development Projects develop parts of the Platform Core and additional components that belong to this building block.
- Non-Core Development Projects develop tools and extension that are not in the Platform Core , but that correspond to the other key building blocks.

The difference is that Core Development Projects will implement research findings on process-oriented learning in a collaborative way mainly in form of open source software development projects, whereas Non-Core Development Projects will implement research findings on process-oriented learning modelling and assessment mainly in form of configuration, adaptation, extensions and add-on implementation of existing commercial software solutions.

Non-Core Development Projects include the configuration of Business Process Modelling Tools because from a technical viewpoint while it may directly upload critical information to the Platform Core, it will begin the communication with the Platform Core thus making the Platform Core the server and the Modelling Tool the client. Learning Scorecard for assessing key performance indicators for learning will even more likely be a selected group of tools that together analyse and visualise the achievements of learning goals.

The actual components that will make up the Platform Core will be detailed in the upcoming Deliverable D2.1 about the Learn PAd architecture. In this document we are only interested on describing the different development approaches for such components, whatever they will be.

2.2.1. Core Development Projects

Core Development Projects are software development projects that implement parts of the Platform Core such as services or components.

As Learn PAd follows an Open Source practices in the Platform Core, each of those projects are seen as full software development projects, meaning that they provide all necessary information including source code, automatic build information, documentation and all necessary configuration files that they can be automatically built and tested. GitHub has been selected to act as the project repository for all Core Development Projects.

There are several reasons why GitHub[1] has been selected:

- It provides a comprehensive platform for collaboratively develop software projects and it's very popular among Open Source practitioners.
- It provides advanced support for managing contribution to the code via pull requests[2] (See Section 4.2.2.1), and code review.
- It provides additional services such as a bug tracking system for supporting software development.

[1] https://github.com/

[2] A pull request is a method of submitting contributions to a software project. It contains the patch representing changes to be made and creates a central place for those changes to be reviewed and discussed. For more information please refer to [1].

- It's free and managed by a third party, relieving us of maintenance effort.

Travis CI[3] has been selected for the continuous integration. The reasons why Travis CI has been selected are:

- It provides a continuous integration environment that is tightly integrated with GitHub, which facilitates the workflow and processes that we are going to use.
- It provides a flexible environment that can accommodate different build and test styles and strategies.
- It's free and managed by a third party, relieving us of maintenance effort.

The aforementioned approach hence addresses the Market and Outreach objective of "releasing of the resulting e-learning platform as open source, and creating an Open Source community around both the core of the Learn PAd platform, and its related components". Core Development Projects, in fact, will follow typical Open Source development practices in order to foster the creation of a community of people interested in Learn PAd that might contribute to it.

As Non-Core Development Projects are outside the Platform Core they follow other practices that are in line with the needs of the partners developing such components.

2.2.2. Non-Core Development Projects

Non-Core Development Projects strongly depend on the nature of the development environment and software platform they are build upon. For the modelling environment, those projects mainly consists of the development of meta models - which are interpreted as configuration files for the meta model platform - and additional extension scripting either using script languages of the corresponding platforms or add-on implementation in form of invocable programs.

For add-on of legacy application those projects can have a huge variety as e.g. a small JavaScript that is added to a Web-Page to establish a connection to the Platform Core, is fundamentally different than an add-on implementation into an ERP solution at the clients side.

In Learn PAd each of those projects are also considered to be a software development projects, although with the restriction of not having a complete software implementation cycle but more likely a configuration, extension or add-on development.

In order to ensure a stable development of the overall Learn PAd platform, although those projects have to fulfill a minimum of documentation, software provision, test configuration and deployment instructions.

[3] https://travis-ci.org/

3 Identified Building Blocks

In order to describe the overall Learn PAd platform and give - for completeness reasons - an overview on the high level building blocks and intended interactions, the next section recaps the functional capabilities described in D1.1 [3]. and discusses indicatively, how it is expected that those building blocks interact with each other.

3.1. High Level Building Blocks

For completeness reasons, we refer to the representation of high level building blocks introduced in D1.1. Figure 1 introduces the four high level building blocks starting with (1) Knowledge, Learning and Business Process Context that considers the complex and heterogeneous operative legacy systems of the end users, (2) Collaborative Business Process and Knowledge Based Learning that enables a process-oriented learning from knowledge workers, (3) Business Process and Knowledge Based Learning Modelling enables the definition of learning processes that are then realized in the aforementioned execution environment and finally (4) Business Process Learning and Knowledge Assessment introduces monitoring and dashboard functionality to identify improvements opportunities.

With respect to software development projects we consider:

1) Knowledge, Learning and Business Process Context as small and proprietary add-on implementation to establish an interaction with the Learn PAd Platform Core,

2) Collaborative Business Process and Knowledge Based Learning will be developed as Core Development Projects and hence will follow the open source paradigm including automated build and testing procedures.

3) Business Process and Knowledge Based Learning Modelling are Non-Core Development Projects that build upon existing tools and platforms and are more expected to be extension projects. As for the definition of meta models and the transformation into executable modelling languages within the corresponding meta modelling platforms there are special guidelines for meta model development.

4) Business Process Learning and Knowledge Assessment are also Non-Core Development Projects that are expected to also include a selection of existing tools that are bundled and configured. Hence, the documentation of those projects also include selection, installation and configuration information to an extent that enable the setup for a demonstration. Alternatively a default setup of one sample scorecard may be realised to ensure a continuously demonstration environment.

3.2. High Level Interfaces

High level interfaces introduce possible interactions between the core building blocks in order to show, how the overall Learn PAd platform is expected to interact. Although the following table is only on high level it is seen as an indicative setup in order to better elaborate the integration strategy that

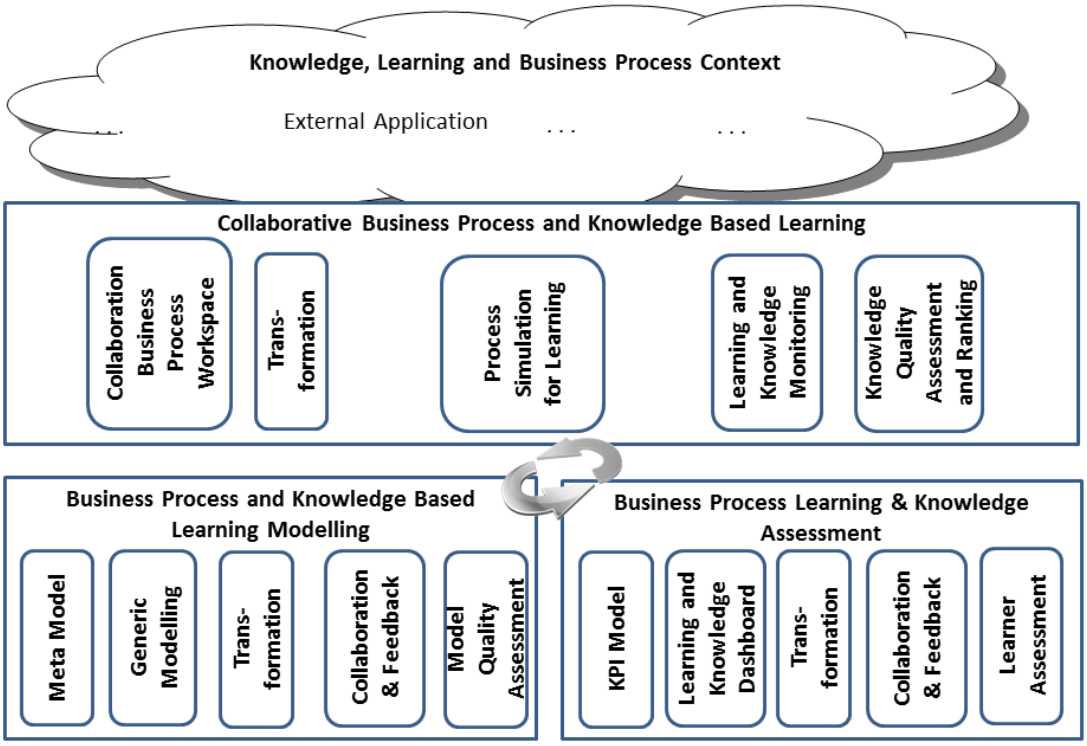


Figure 3.1: High Level Building Blocks for Learn PAd Architecture

is explained in this document. Improvements on the interaction are foreseen in the corresponding architecture deliverable D2.1.

Interaction between	Interaction type
Core - Legacy Application	The interaction will be performed on a case by case basis, ranging from a simple hyperlink - realising a ui interaction-, to the invocation of some add-on scripts - realising a functional interaction, till the query and data access of e.g. KPI data, which will perform a data interaction.
Core - Modelling	The interaction will be based on a data exchange of the business process model information. The user interface and the functionality will be autonomous. It may be the case that track changes in model need further interaction, hence a small functional integration may be necessary.
Core - Assessment	The interaction will be based on a data exchange from the Platform Core towards the assessment building block. The analysis and visual representation will be autonomous in form of applications that realise this building block.

4 Software Development Processes

In this section we will describe how partners are going to collaborate around the development activities that will lead to the release of expected software deliverables.

4.1. Collaboration Processes

To have a fruitful collaboration we must define some best practices that should be followed so that it is clear where to find information, and how to come up with final and agreed content.

This is true in every phase of the development of the Learn PAd project, from the discussion that happens between the partners, the definition of the architecture of the system, its design, to its final implementation and documentation.

Since the consortium is made of several partners with different background, we don't want to be very strict on these practices (e.g., we don't want to regulate the collaboration down to the code style that should be used in the implementation of the components).

However we consider that there should be some common "synchronization place" where the global knowledge of the project should be kept in an up to date state.

We expect partners to collaborate over the usual means - mostly discussions on the mailing lists, conference calls, direct face-to-face meetings. Once a subject under discussion is finalized, it should be re-elaborated by the person who started the conversation and stored in this "synchronization place".

This has already been done so far - for example, in the wiki each WorkPackage has its own page with the relevant informations about the activities carried on in its context (e.g, `http://wiki.learnpad.eu/LearnPAdWiki/bin/view/WP1/WebHome`)

We expect this method of collaboration to continue in the future, with a particular attention to the aspects described in the following sections.

The collaboration processes will leverage the infrastructure described in Section 5.1 in order to allow people to effectively collaborate and where the portal (See: Section 5.2.1) will play a central role.

4.1.1. Architecture

For Architecture, as mentioned in the Description of Work, the "4+1 architectural view model" [2] will be used. This implies the use of diagrams and graphical artifacts for describing the different views on the architecture (e.g., logical, physical, etc.). Current activities in WP2 are already using the UML language in order to model architecture. This allowed partners to have a centralized repository that allows to store and edit them concurrently. Once the collaboration around architecture reaches a stable state, this state will be reflected in the wiki in a corresponding page which thus will always contain the current stable state of the architecture.

4.1.2. Interfaces

For Component's Design we differentiate two kind of interfaces that must be thoroughly documented: API and UI (where applicable). It is important to freeze the discussions about how a component will

expose its functionalities in order to guide the implementation more effectively.

In order to document the API the design document must clearly describe what are the endpoints of the service, the protocol used, the input and the output and the error conditions that can happen when interacting with it.

Since there are different ways of implementing a component API that go from a library to a remote web service, different types of documents might be used:

- Documenting the API of a library can be done using an annotated class diagram. If the component will be implemented in Java the JavaDoc resulting from the interface specification can be a viable way.
- Documenting a RESTful API can be done using a template like the one used for documenting Twitter APIs used[1]
- Documenting a WebService API can be done by publishing an annotated version of the corresponding WSDL

For those components that will also provide a UI it is important to provide graphical mockups of these interfaces and to describe the possible transitions between the different screens. These mockups can be produced by using several drawing tools[2]. We do not prescribe any specific tools to use, as long as a clear wireframe mockup is exported and included/attached in the corresponding design document.

We don't expect to have a waterfall approach to software development where these specifications are set in stone once and for all, however it must be clear that at every development cycle (see next sections) a clear design document about what will be implemented must be provided in the wiki under a well identified space DesignDocuments. An index page will provide a browsable directory of all the current documents available.

4.1.3. Configuration and Deployment

Each component must have up-to-date information about its configuration methods and deployment instructions. In order to do so a document for each component will be available in the Components space in the wiki where people involved in the development of the component will collaborate in order to provide this information readily. These pages will also be referenced in the actual source code of the component so that they can be accessed and found also from the source code repositories (see next sections).

4.2. Development Processes

The requirements and objectives of each components will be first detailed in the relevant deliverables, and then they will be added, in the form of items in the provided Bug Tracking infrastructure (See: Section 5.3.2). In this way progress on the implementation of each requirement can be tracked.

These items will be used by Partners responsible of a component to define a roadmap that details objectives and timeframes for addressing them. Once deliverables have been delivered and the corresponding items on the Bug Tracker marked as "fixed", then new issues may be created for newly discovered requirements.

The Development Project Portal (See Section 5.2.1), which will be based on our project wiki, will also provide a common entry point for storing information about the artifacts created and manipulated by these processes (i.e., source code, roadmaps, etc.)

[1] https://dev.twitter.com/docs/api/1.1/get/statuses/retweets/%3Aid

[2] An example of such a tool is https://moqups.com

4.2.1. Development Roadmap

The partner roadmap will detail timeframes and objectives for the components for which this partner is responsible. At any given time, a partner roadmap should contain goals for the next three months, linked to corresponding items in the bug tracking system. Development milestones will take place every two weeks and will consist of a brief roadmap meeting to check in on progress and set new goals.

Each partner's roadmap should be contained in a document on the wiki in the Roadmaps space where it can be reviewed at each bi-weekly roadmap meeting and each partner will be responsible for setting their own objectives. The global roadmap will be collaboratively developed and will contain a broad picture of the project, aligned on the deliverable deadlines indicated in the Description of Work.

4.2.2. Core Development Projects

The Agile Methodology[3] prescribes architectural design should not be set in stone and changes can be made as experience is gained, following with that thinking it will be acceptable for Core Development Projects to be designed and developed simultaneously, starting out unstable and stabilizing over time.

Discussion of architectural design should take place on the mailing list, chatroom or other means where history of the conversation can be recorded so as to provide a record of the thoughts and reasoning. As ideas solidify and consensus forms, they should move to the wiki where they become official documentation.

Source Code Management Guidelines

All source code, binary code and configuration needed to build a working implementation of the Platform Core will reside in a public GitHub, repository. Each developers will work on his own copy of the repository (known as a fork[4]) and propose new functionality or bug fixes using GitHub pull requests which integrators[5] will have the duty to review. An integrator may accept the pull request, request changes or further information or in an extreme case, reject it pending further discussion. Despite having direct access to the main repository, an integrator who is also make sure to use pull requests for integrating their own functionality so as to allow for discussion with other integrators and developers and to keep record of the contribution.

Since all source code will be stored in a single GitHub repository, each developer will have a working copy of the entire project. To simplify collaboration, each component will be contained in one subdirectory within the source structure. The partner in charge of managing development of a component will be free to organize the content of this subdirectory in any way as long as the following standard files are contained within:

- `readme.md`. This file will contain the entry-point to all documentation for the component. At the very least, this file must contain contact information for the manager of the component, a list of the authors, a high level description, a link to their roadmap (See: Section 4.2.1) and a link to a location containing extensive documentation for deploying and using the component.
- `build` This will be a file containing executable script code which can automatically run on the target environment (See: Section 6.2). The objective of this build script is to do whatever automated operations are necessary (such as compiling and packaging) to produce a working component.
- `builddeps.txt` This optional file will contain a list of applications which need to be installed using the target environment's package manager[6] in order for the component to be built.

[3] See `http://agilemethodology.org`

[4] See `https://guides.github.com/activities/forking` for more information about the GitHub workflow.

[5] Developers of the Platform Core will be responsible for integrating changes to core components.

[6] The target environment's package manager is an application which can automatically download and install software on that system.

- `out` This folder will be created by the build script and will contain the result of the build, everything that is necessary for this component to run.
- `out/etc` This folder, named after the UNIX standard /etc configuration folder will contain all of the configuration files necessary for the component.
- `out/start` This will be a script file which initiates the completed component, it will expect it's configuration to reside in the out/etc folder and must be capable of starting the component with no manual intervention.
- `out/stop` This script will behave similarly to start but will stop the running component.
- `out/rundeps.txt` This optional file will contain a list of application names, like builddeps.txt, but with applications necessary for the component to run.

A special component shall exist for all tests which require the Platform Core to be functioning. The duty of this component is to produce a test report and upload that report to the wiki where it can be reviewed.

Development Process Recommendation

All developers should adopt the check in early and often development practice[7]. Code in development is not expected to be bug-free but great importance is placed on collaboration. In a fast-changing collaborative software development project, frequent pull requests help everybody. They help the developer making them because he is able to avoid rewriting code to cope with changes in other elements of the system and they also help other developers who are able to see and adapt to changes in this developer's component.

Developers are encouraged to create automated tests for their components and such automated tests shall be started by the build script (See: Section 4.2.2.1). Integration and Functional Tests shall be developed within the testing component but the exact testing framework or frameworks to be used shall be discussed later.

Documentation Guidelines

Documentation for a Core Development Projects shall be either in the source repository or in the provided wiki. In either case the readme.md file in the source repository shall act as an entry point, containing URLs which allow the reader to navigate to all relevant documentation for the corresponding component - these URLs will mostly point to information accessible via the Development Project Portal (See Section 5.2.1) At the very least, the readme.md file shall contain contact information for the partner responsible for the component and the URL of their partner roadmap on the wiki.

In the case of communications protocols and application programming interfaces between two or more components in Core Development Projects or between a component in a Core Project and one in Non-Core Development Projects, or other programs external to the Learn PAd project, the details of the protocol or interface must be documented. If any configuration is needed to customize a component for different deployment scenarios, this configuration must also be documented. Finally, the objective of the component, how it works and how to collaborate on it's development (if applicable) should be documented.

[7] Check-in early Check-in often refers to the practice of merging changes to code with other developers at short time intervals to minimize divergence. See http://blog.codinghorror.com/check-in-early-check-in-often

4.2.3. Non-Core Development Projects

In order to organise all development projects, also those who are not dealing with components in Core Development Projects, the Non-Core Development Projects also provide guidelines for a safe software development.

Software Management Guidelines

Similar to the Core Development Projects there is a structure to provide the necessary software and corresponding configuration files. All applications and configuration needed to interact with the Platform Core have to be accessible either in form of binaries that can be installed on the demonstrator server, or in form of a Web-Services that provide the required functionality toward the Platform Core.

All relevant documents, files or links are stored in an Subversion directory that is accessible from the development project portal (See: Section 5.2.1). Each developer will work on his own copy of the repository and propose new functionality or bug fixes using its individual reporting and bug tracking system. Only changes that influence the interaction with the core system, need further elaboration. This is mainly the case for the data schema that correspond to the developed meta model, as well as possible functional interaction.

- `readme.md` This file will contain the entry-point to all documentation for the component. At the very least, this file must contain contact information for the manager of the component, a list of the authors, a high level description, a link to their roadmap and a link to a location containing extensive documentation for deploying and using the component.
- `build` This will be a file containing contact points, installation descriptions and documentation for building the system. In case the non-core component is not provided as binaries but as a running service, this directory contains files describing the access to this service.

It is highly recommended that a directory shall exist for tests in appropriate form, which require the component to be functioning. The duty of this component is to produce an appropriate report and upload that report to the wiki where it can be reviewed.

Development Process Recommendation

As the non-core development projects are individually treated the overall recommendation is only limited to the issues that either affect the overall development progress or the development of other components.

Stable versions should be uploaded either as binaries or executed as services in reasonable intermediate but stable versions, to allow other components to witness the development progress but always rely on a stable accessible version. The separation of development versions, internal test and demonstration version and project wide demonstration is recommended as one way of providing periodically stable services.

Developer are encouraged to create automated tests of the providing or consuming side towards or from the core components as soon as possible in order to periodically check, if the development of their own environment does not corrupt already implemented code from other components. In addition to test towards other components, it is recommended to have automated tests that observe the own development progress.

Documentation Guidelines

Documentation for a Non-Core Development Projects shall be in the provided wiki in form of an entry page. Detailled information may be pointed to necessary documentation files, including modelling tools.

Additionally the readme.md file in the software repository shall act as a collection, containing URLs which allow the reader to navigate to all relevant documentation for the component. At the very least, the readme.md file shall contain contact information for the partner responsible for the component and the URL of their partner roadmap on the wiki.

Similar to Core Development Projects, in the case of communications protocols and application programming interfaces between a component in a Core Project and another in a Non-Core Development Projects, the details of the protocol or interface must be documented.

4.3. Integration and Testing Processes

In order to enable an integrated system, the overall aim is to quickly produce integrated prototypes, although those initial versions will consist of dummy proxy services, which are exchanged step by step through full fletched functionality. A minimum of agreed integration and testing actions are provided. We also point out that what is presented in this section is a starting point. In fact testing processes will be refined and improved throughout the project and presented in further deliverables (i.e. *D7.3 : Integration Testing Procedures*, and *D7.4: Integration Testing Procedures – Final Iteration*)

4.3.1. Build Process

There are two ways to create a running and accessible software depending, if it is a core component that has been implemented following the open source approach, or it is a non-core component that builds on existing closed source platforms. In case of the latter the provided recommendations are minimum requirements.

Core Development Projects

Each Core Project will be required to automatically build and run in the target environment. Applications on which a component depends in order to build shall be named in builddeps.txt and applications upon which the component depends in order to run shall be named in rundeps.txt as defined in Section 4.2.2.1. The building of the Core Components shall be carried out by a main build script that will install the applications needed for the build of each component, execute the build script for that component, then aggregate the results of the build into a single application constituting the Platform Core.

In the event that a component is unable to build or a build time test fails, the build script shall use a non-zero exit code in order to communicate to the main build script that the build has failed.

Non Core Development Projects

Each Non-Core Project will be required to either provide all necessary information for a manual or automatic build, or provide a running service that can be accessed. The building of the Non-Core Development Projects will be carried out mainly by either following a certain procedure on installation and deployment steps, or by accessing a running service.

In the event that a component is unable to be built or can be accessed, a contact person must be available to provide technical support.

4.3.2. Integration Process

The philosophy is to regularly exchange one stable version of software application with another stable version. Hence, mechanisms are discussed that ease the Continuous Integration[8] (see Section 6.2).

[8]Continuous Integration is a software engineering best practice as defined in http://en.wikipedia.org/wiki/Continuous_integration

Core Development Projects

The integration of all components in Core Development Projects with each other and with the Platform Core will be automated by way of the main build script described above in Section 4.3.1.2. Integration of any Core Component with an external service will be the responsibility of the developer(s) of that component and since the components will be developed together and continuously built, tested and deployed on a live testing server, we foresee few issues with integration.

Non Core Development Projects

Integration of a component in a Non-Core Project with another one in a Core Project will be the responsibility of the developer(s) of that component. A timely communication, ideally following agreed development and integration roadmaps should ensure that the new version that is going to be integrated, does not break or use outdated functionality due to unsynchronized integration.

The live testing server for the Platform Core will allow Integration Testing of Non-Core Development Projects with the current state of the Platform Core.

4.3.3. Testing Process

The testing process is distinguished in an internal part of the building block and a interaction test between the building blocks.

Internal testing procedures will follow the common practice of those development teams, hence there will be different testing approaches for Core and Non-Core Development Projects, as well as between Non-Core Development Projects as each of the teams have their own experiences and requirements. At this state, regression, stress and load balancing tests are highly recommended, but it is in the responsibility of each development team to timely provide stable versions of the software.

Testing between building blocks, follows the indicative high level interaction introduced in Section 4.1.2 and is adapted according to their evolution. In case of user interface interaction, tests are recommended that test the click-stream of user interfaces, in case of functional and data interactions, regression, stress and load tests are recommended. Hence the aim of the integrator is, to set up test procedures with the help of the individual development teams and continuously run the tests with alerting in case of conflicts between building blocks.

4.4. Demonstration Processes

The aim is to establish as quick as possible a demonstration scenario that explains in text, slides, mockups or videos the aim of the Learn PAd platform. In parallel this demonstration package is supported by a demonstration environment that in a stepwise approach provides stable running prototypes that continuously exchange parts of the mockups with real software.

The idea is to communicate a coherent demonstration story from the beginning and subsequently exchange parts of the story with real running software. At this stage it has to be stated that media breaks from the mockup version of the demonstration to the real version are not considered to be an issue, as integrating real software also into the mockup version might on some part be feasible and easy but in other part require an unnecessary effort that does not provide much clearance.

The mockup version and the integrated prototypes shall be seen as running in parallel. As the real prototype gets more functionality, the mockup will be replaced piece by piece.

A Common entry point for both (a) the demonstration story including slides, videos, pictures, business process models and the like as well as (b) the demonstration of the system either in the form of a mockup or as a running prototype is foreseen as a Wiki page, enabling demonstration of the Learn PAd system at any time independent of the current integration of the software.

5 Development Infrastructure

A set of infrastructure tools will support the development of the Learn PAd Core Development Projects as well as the Non-Core Development Projects. Selection of appropriate tools is important in order to foster close and efficient collaboration, ensure that changes and decisions are visible to everyone involved and minimize the effort required for the most common tasks throughout the project.

5.1. Collaboration Infrastructure

The main objectives of collaborative software development are to agree on methodologies as efficiently as possible, to keep a historical record of the decisions made and their reasoning and to synchronize rapid development and prototyping of interacting components developed by multiple parties. While techniques such as meetings, and telephone conference calls are useful, text based collaboration is ideal from a historical standpoint because it can be recorded and searched and can often be converted to official documentation with minimal effort.

5.1.1. Mailing List

An email mailing list has already been provided by CNR and is available for general decisions about the project. This will allow for discussion about general project management issues, but also for technical aspects of the upcoming deliverables.

In particular the mailing lists will be used to facilitate discussions about architecture, design and technical aspect addressed by the partners during the project evolution.

These discussions, once they're over, will be summarized in the corresponding tools used for keeping track of the current state of the project (e.g., wiki pages, architectural models and bug/tasks)

5.1.2. Architecture Modeling

Learn PAd architecture is an important aspect that will guide all the partners throughout the project. In order to keep architecture models in a consistent state we will use specific tools that will allow users to collaborate around these artifacts. In particular, in the context of WP2 activities, we've set up and used for this purpose UML model collaboration server. This allowed us to work all together and in a centralized way on the architecture and all the related artifacts, and keep track of the changes.

5.1.3. Documentation

The documentation will be accessible through the Development Projects Portal (See Section 5.2.1) which will be based on the XWiki instance that is currently available on `http://wiki.learnpad.eu/`. This allows partners to create and organize the documentation for their components, internally track development progress and keep up to date on past and upcoming deliverables, but also to collaborate around work in progress documents.

5.2. Development Infrastructure

Actual software development will take place in a set of Source Code Management Systems. Core Development Projects will be placed in a repository hosted at GitHub while the developers of each Non-Core Project will be responsible for determining where and how it will be hosted and their decisions will be expressed on the Development Projects Portal (See: Section 5.2.1).

5.2.1. Development Projects Portal

The Development Projects Portal will be organized as a set of pages hosted in the wiki (See: Section 5.1.3). These page will provide an entry-point to the documentation on each component, explaining where it is hosted as well as how to use, test and integrate it with the overall project.

5.2.2. Source Code Repository

The source code for building the Core Development Projects will use a common Source Code Repository, as described in Section 4.2.2. This repository will be hosted at GitHub, a popular venue which provides free code hosting and collaboration tools to open source projects. Non-Core Development Projects do not share source code and have therefore individual infrastructures.

5.3. Integration and Testing Infrastructure

It is critical to development of both the Core Components and the Non-Core Components that the Core Components are continuously integrated, tested and made available on a live testing server for automated and manual integration testing of the entire Learn PAd system.

5.3.1. Continuous Integration

Continuous integration is an automated process by which the current state of the source code is built into a final product, any automated tests are run and the final product is made available to developers so that they can test the interaction of changes they made to the code in near-real-time.

As already described in Section 4.3.2, Continuous integration will be provided by Travis CI, a hosted, distributed continuous integration service used to build and test projects hosted at GitHub. Like GitHub, Travis CI service is free for public projects and Travis CI provides email notification of build failures and GitHub integration for automatically testing pull requests before they are accepted. The Operating System on Travis CI build machines is Ubuntu Linux 12.04 LTS[1] which will be referred to as the target environment.

Every time the Continuous Integration server builds the Platform Core, the completed platform will be deployed on a live server which uses the target environment so that it can be accessible to all of the collaborators. This include automatic deployment for the Platform Core and manual deployment for the Non-Platform Core.

5.3.2. Bug Tracking

The bug tracker provided by GitHub as part of the Core Development Projects repository will be available for reporting defects, feature requests or requirements, and other miscellaneous to-do items on both Core Development Projects and Non-Core Development Projects alike.

[1] Ubuntu 12.04 LTS is the stable version of a popular server operating system. Software which runs in this environment can be expected to run in many popular server environments with little to no porting effort.

5.4. Demonstration Infrastructure

In order to realize the demonstration scenarios that have been defined (See: Section 4.4), a set of servers will be provisioned so that software can be deployed on them and be accessible for all the members of the consortium to actually use and test the developed solution.

These servers will be provisioned and managed accordingly to the nature of the components that will be deployed on them.

In principle, all of them might be provisioned on the same infrastructure (e.g., the Regione Marche Cloud), however depending on the specificities of the components that will be deployed on them, it might be possible that some partners will need to have total control of these servers. In this case they will handle the provisioning and management and ensure that they are accessible to authorized users.

For Core Development Projects the server will host a snapshot of the Platform Core that is used for integration testing on the live server. These snapshots will be taken at given checkpoints, according to the roadmap and the project's milestone, when the state of the components will be declared stable (i.e., the scheduled features have been implemented, tested, integrated, and there are no critical bugs that affects them) The idea is to automate this process of deploying a stable version of the components on the demonstration server as much as possible.

Non-Core Project teams will use similar guidelines, providing a snapshot of their development environment once the components reach a stable status and become mature (i.e., the scheduled features have been implemented, tested, integrated, and there are no critical bugs that affects them)

The demonstration infrastructure, finally, will enable the deployment of the integrated Learn PAd platform at different intermediate checkpoints so that it is always ready for carrying on demonstration. This is particularly important to minimize the risks of not having a running and integrated software at the major milestones indicated in the Description of Work.

6 Integration Plan

In this section we will describe the integration plan that we foresee for the duration of the project. Many of the practices have been described before and are thought to minimize the risk of integration problems just before official deadlines identified in the Description of Work.

In the following sections we detail how we are planning to integrate all the components of the system following a plan based on what is described above.

6.1. Workplan Based Integration

In Section 4.2.1 we described how each partner must provide a roadmap for documenting the progress of the work he is doing with respect to all the components he is responsible for.

These roadmaps shall be used in order to define global roadmap that will contain a broad picture of the software development aligned with the necessary deadlines that are required for the overall performance of the project. The process of integration will begin with the Continuous Integration of the Platform Core and the deployment on the live server. The developers of the Core Development Projects will be expected to write automated integration tests and, prior to releases, performing manual testing for verifying the Core Development Projects integrate with one another. Developers of Non-Core Development Projects will be able to continuously integrate and test their work against the Core Development Projects using the live server and may provide testing stubs to the Core Project developers in order to facilitate protocol compliance unit testing.

Having such an integration plan will allow the consortium to periodically verify that the system being developed is in a sane state and can be officially released as a deliverable.

6.2. Continuous Integration

In Section 4.3 and Section 5.3, we described continuous integration as a cornerstone of the development process, and our aim at automate it as much as possible.

Continuous integration will provide thus a view of what problems are in the system under development, either because there is something that is not building correctly or because tests are failing.

This information is precious feedback for the integration plan because it will allow us to reorganize the roadmaps accordingly.

6.3. Release Process

Releases will take place once every 3 months with exact release dates being set at the previous roadmap meeting. Figure 6.1 shows the 3 months intervals where releases are expected, and gives also an overview of which releases coincide with actual software deliverables that are expected as stated in the Description of Work.

The partner responsible of a project, whether of a Core Project or Non-Core Project will be expected to contribute to a Release Notes wiki page in the ReleaseNotes space, reporting on any new features

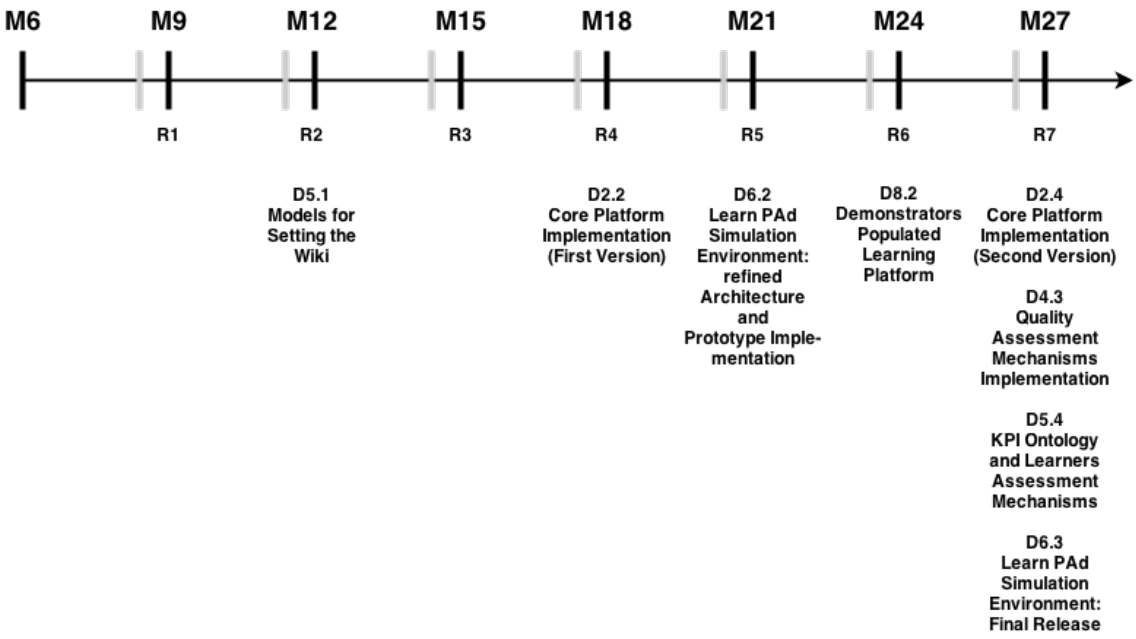


Figure 6.1: Releases Timeline

and bug fix in components for which they are responsible. In the 2 week period prior to the release, developers will be expected to verify the functionality of the components for which they are responsible and the integrators will only accept pull requests that fix important bugs and are regarded as low risk to the platform's stability. The actual release will comprise creating a tag in the code repository, making the packaged Platform Core available for download and linking to it from the Development Projects Portal (See Section 5.2.1), then deploying the Platform Core on the demonstration server. In concert with this, developers of Non-Core Development Projects will need to release their projects, adding binaries to the Development Projects Portal where applicable and synchronizing their demonstration infrastructure accordingly.

7 Summary and Conclusion

In this deliverable we provided an initial description of some best practices and tools that will facilitate a process of developing the software deliverables that are required by the project. This should provide an initial framework to all partners for starting to work on the artifacts needed for building the system that will be delivered by the project.

An integration plan based on well defined check-points has also been proposed in order to minimize the risk of not having an integrated software deliverable at the major milestones.

Since we are working on a challenging project in a very heterogeneous consortium, we are aware that some of what is described above might evolve and be refined. Should this be needed, an amended version of this document will be released in order to reflect these changes.

Bibliography

[1] GitHub Inc. *Using pull requests*. Available at : https://help.github.com/articles/using-pull-requests – last access : 27 July 2014.

[2] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50, 1995.

[3] Robert Woitsch, editor. *Requirements Report*. Number Del. D1.1. The Learn PAd Consortium, 2014.