Co-funded by the
Erasmus+ Programme
of the European Union

TEAL2.O
TECHNOLOGY ENABLED ACTIVE LEARNING

TEAL2.O ORGANIZATIONAL REQUIREMENTS

CONTENTS

This document clarifies the organizational requirements for the TEAL2.O development process. The intended audience is the developer teams.

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

DEVELOPMENT TOOLS

Developer teams are required to use the following development tools:

Git

Installing Git on your computer and setting up can be found here. It will be used for version control.

Apache + PHP

Based on your system, you can install Apache2 and PHP.

- Debian: LAMP
- OS X: Mountain Lion Server
- Windows: Apache2 + PHP
- PHP Extensions and libraries
- Enable SSL for Apache (Transitioning to HTTPS)

More information: Installation

Database

Based on you system, setup MySQL Database for Moodle.

Moodle

Manual installation: Quick guide

✓ Moodle Development Kit (MDK)

Only for Linux and MacOS (not available for Windows).

- Setup
- Sample to create some instances

mdk create --install --engine mysqli --run mindev users makecourse -n stable_master-mysql

✓ Docker

Docker containers will be used to allow developers to wrap up an application with all essential parts, i.e. libraries, dependencies, and ship it all out as one package. By doing this, the developer can be assured that the application will run on any machine, regardless of any customized settings. Containers also work in isolation from each other allowing a range of tasks to occur independently.

✓ IDE

You can setup any of the following IDE and use it for development:

- Sublime

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- PhpStorm - Setting_up_PhpStorm for Moodle development.
- Eclipse
- Netbeans
- ViM
    - ✓ Detailed Moodle Setup

More details about Moodle development environment setup can be found HERE.

There is a YouTube channel that shows clips with guides and tips for the Moodle setup.

>> Back to contents

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

SUPPORT AND COLLABORATION TOOLS

In a distributed environment, the building of a significantly large platform requires a lot of collaboration and consistent feedback exchange among participants. In the environment needed to create TEAL2.O work is distributed across several regions and several institutions within each region. This setup requires active collaboration during the design and implementation processes. This will create more engagement among all parties and we will get the best out of all developers, while also applying the latest trends, best practices and technology choices that will contribute to the overall quality of the project and will increase the motivation of the participants.

The types of collaborative tools that will be employed in the TEAL2.O development process are:

Documentation and Task Allocation

Cloud-based **Atlassian Confluence** will be used as it is easy to sign up and it is using matured Wiki-style documentation with concurrent editing enabled. It is Cloud-based, so no maintenance is required.

Within individual teams, it is recommended to use **JIRA** for specific task allocation.

Delivery Methodology

It is recommended to start small and iterate through rather than trying to design everything upfront. It is also absolutely important for all stakeholders to be kept informed of the development of the platform and all tracking towards their final vision.

Therefore, the developer teams will use Agile, especially Scrum, methodology, with two-week Sprint across all teams, with Sprint start and end dates aligned.

Source Control

All source code needs to be maintained in a collection of repositories, so that it is accessible by all participants and everyone gets to contribute towards high-quality code and best coding practices. It is also important that all code commits are reviewed by at least another person so there is less chance of human error and so that coding consistency is maintained throughout the platform.

We will keep at least two branches of source code. One branch will be used for the stable version and the other one will be for the daily build. The stable version will be maintained by a person after the review of the current version at the end of sprint or per period.

Therefore, the developer teams will use Git-based Project for the entire platform and repositories for each domain/component/feature.

Continuous Integration and Continuous Deployment (CICD)

All developed code needs to be continuously integrated with the application server and database and deployed in an environment that is the same as the final environment. This provides early insight into any software bugs and required changes and reduces the risk of deviation from the ultimate vision.

Therefore, the developer teams will use cloud-based CICD tooling. It does not require any management other than setting up your own pipelines.

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Development Environment

A development environment needs to be dedicated to each component development team so they can continuously develop and test their code without interfering with other teams' development activities.

Therefore, the developer teams will use cloud-based contained environment for each component development team, as well as managed roles and groups that are respectively associated with each component and team. This will give flexibility for a team to be involved in one or many components.

Component Test Environment

The component test environments are needed to make sure any component can be deployed and tested in the same way as it is developed, and to make sure that there are no environment-related issues present before it gets promoted to next level for an end-to-end testing.

Therefore, the developer teams will use cloud-based contained environment per each component across the platform, so this is always kept functional and updated during the development process.

Integrated Environment

This is the final environment where all components expected to be integrated and all the completed components of the entire platform are always kept functional. It is expected that this environment will act as a gluing environment over the component environments unless a copy of component environments is required for different types of testing such as user acceptance, partnership testing, performance testing, etc.

Therefore, the developer teams will use cloud-based contained environment with specific roles assigned for each type of activity and these roles may or may not be assumed by the developers who developed the component. It is expected that most often a nominated group of people will manage this (instead of the original developers) so it is considered from the user point of view.

>> Back to contents

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The main external constraint for the development teams is the need to comply with Moodle standards and documentation. This is absolutely necessary as the idea is to develop all features as Moodle plugins. This will improve their usability and user acceptance. Developer teams will be required to comply with these constraints.

Although Moodle is open source and we can change anything in Moodle to support our needs, the best and most sustainable way to extend it is to write a plugin (sometimes called "a module"). This allows us to incorporate Moodle updates and we will only need to maintain and update the plugins we have developed. It should be reiterated here that the purpose of ensuring Moodle integration is to allow users who already use Moodle as their LMS to incorporate the TEAL2.O features through Moodle in a very convenient way.

Moodle supports a wide range of plugin types. It supports standard and advanced (admin) plugins. The complete list of plugins type details can be found at Plugin types.

Moodle offers a thorough documentation for development. Developer documentation can be found here. It has information on several types of API that are needed to connect with its core and other external systems. These are essential when writing Moodle plugins.

There are several guidelines to follow when developing Moodle plugins. They can be found in dev doc. In Moodle dev doc, they have listed several development tools that are necessary and/or useful. Overview of the communication between Moodle components can be found here.

☐ Standard plugins

Moodle has a general philosophy of modularity. There are nearly 30 different standard types of plugins and even more sub-plugin types. However, all of these plugin types work in the same way. Blocks and activities are the only small exceptions.

See Moodle plugins and Moodle sub-plugins for more information.

☐ Local plugins

The recommended way to add new functionality to Moodle is to create a new standard plugin (module, block, auth, enrol, etc.).The local plugins are mostly suitable for things that do not fit standard plugins.

☐ Custom/ local plugins

- Local plugins are used in cases when no standard plugin fits, examples are:
- event consumers communicating with external systems
- custom definitions of web services and external functions
- applications that extend Moodle at the system level (hub server, amos server, etc.)
- new database tables used in core hacks (discouraged)
- new capability definitions used in core hacks

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- custom admin settings
- extending the navigation block with custom menus.

List of differences from normal plugins:

- always executed last during install/upgrade - guaranteed by order of plugins in `get_plugin_types()`
- are expected to use event handlers - events are intended for communication core-->plugins only, local plugins are the best candidates for event handlers
- can add admin settings to any settings page - loaded last when constructing admin tree
- do not need to have any UI - other plugins are usually visible somewhere.

☐ Coding Guidelines

Developer teams should become thoroughly familiar with Moodle's: coding guidelines. These guidelines should be strictly followed.

☐ Tutorial

There is a Tutorial to help you learn how to write plugins for Moodle from start to finish, while showing you how to navigate the most important developer documentation along the way.

☐ Moodle 3.9 Database

Full list of tables in Moodle database can be found here.

Moodle Developer Video Tutorial can be accessed from: YouTube channel. It shows clips with guides and tips for the Moodle development.

>> Back to contents

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.