Adapt Learning: Authoring Tool Concept and Vision

Document control

Abstract:	Describes the how responsive applies to the Adapt Learning Project		
Author:	Sven Laux	Version: 0.1	Date: 07 / 11 / 2013

Summary of Changes:	**Versions**	**Date**	**Description**
	0.1	07 / 10 / 2013	Initial draft for review. To be turned into a content page for the community site.

DOCUMENT CONTROL 1

PURPOSE OF DOCUMENT 3

CONCEPT OVERVIEW DIAGRAM 4

EXPLANATION 5

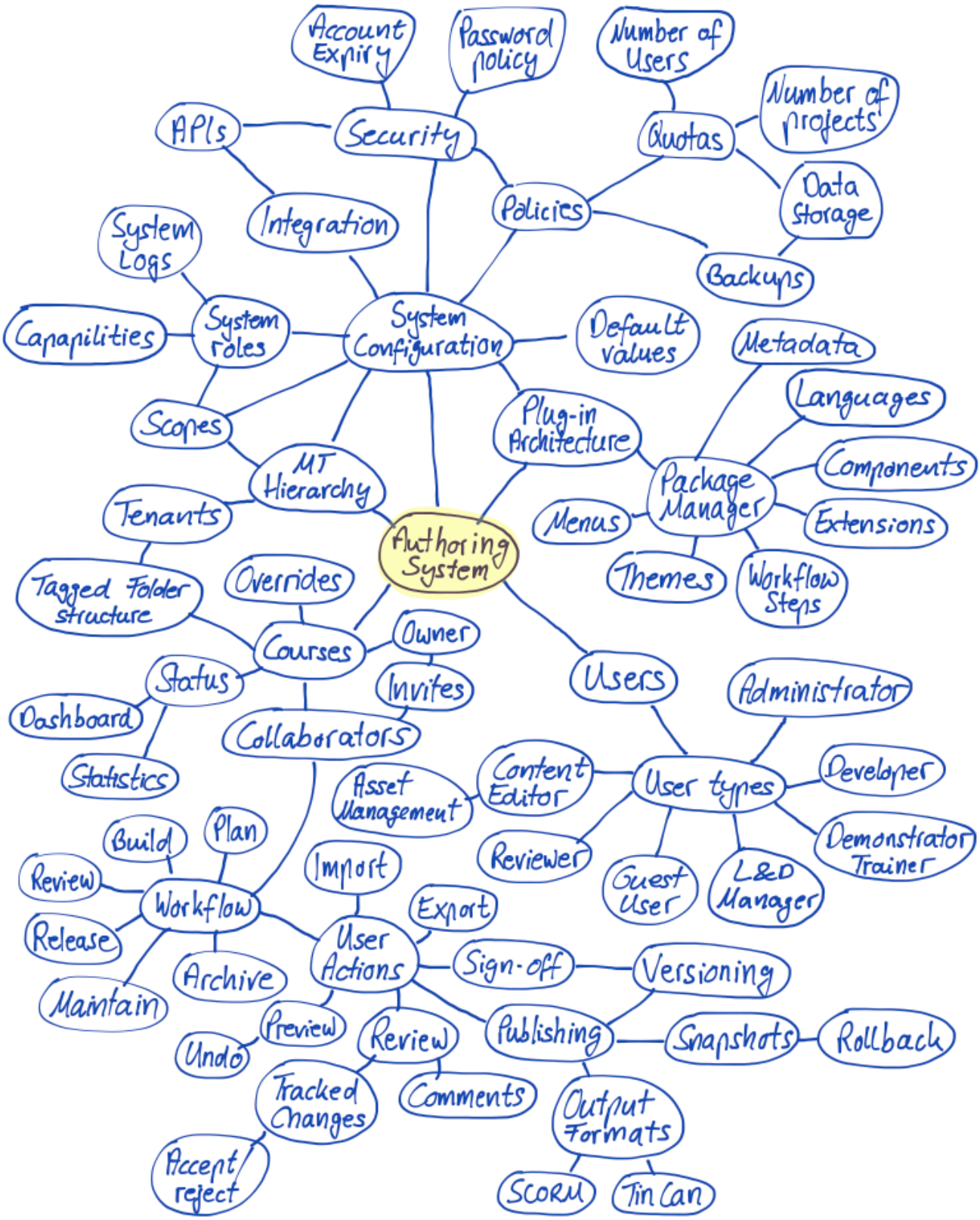
Purpose of document

The purpose of this document is to outline the vision and concept of the Adapt Learning Authoring Tool. It contains a concept diagram, similar of a mind map of the key functionality, components and elements of the authoring tool.

This document is not a specification document. It is intended to help the project team and wider community understand the full product we are aiming for. As such, the document will therefore set the context in discussions about requirements, system architecture, specification etc.

The document is also intended to help newcomers to the project to get an overview.

Concept overview diagram



Explanation

ID	Node title	Description
	Authoring System	Central node for visual navigation
	System configuration	Heading / major system domain. Thinking about the overall system configuration is key. Our aim is to build a flexible and highly configurable tool yet stay focused on ease of use and creating an intuitive tool for non-technical end users.
	System roles	System roles deal with the permissions users have to carry out tasks in the system. We have been guided by Moodle's approach of system roles. This means that system roles are essentially collections of capabilities (see below), which apply to different system scopes (also below). There are predefined system roles (see user types below) but is it also possible for system administrators to create new system roles.
	Capabilities	Individual user actions, which can be carried out within the system. This allows very granular control over system roles.
	Scopes	The major areas for which roles and capabilities apply. For example: • System wide • Tenant wide • Project / course wide • Etc.
	System logs	An audit trail of user actions taken in the system, which can be by administrators used for handling users support issues, fault finding etc.
	Integration	A heading prompting us to think in terms of facilitating integration with other systems in the future without having to hack the core code and architecture.
	APIs	Integration points for external systems. These will likely be implemented as web services.
	Policies	Part of system configuration and aimed at system administrators. There are several sub components, which deal with the administration of resources, quotas and security. A server based authoring tool depends on an underlying hosting infrastructure with limited resources. These require management in a similar sense to web hosting control panels. Unlike a desktop-based tool, a hosted system incurs cost by default but is also key to commercial service-based business models developing later on (when the open source project matures). Examples are Moodle, Linux etc. Policies are a common vehicle to enable administrators to express and control resources, usage and security among other items.
	Security	Heading to help explore and organize configuration settings.
	Account expiry	Account expiry is a feature often requested for enterprise systems. The

		authoring tool should therefore enable automated processing of account expiry, including the configuration of rules and events which lead to user accounts expiring and what happens when they do.
	Password policy	The authoring tool must have the ability to enforce rules that determine the strength of passwords and their expiry.
	Quotas	A general heading / node dealing with the resources tenants and users are allowed to use. Rather than a web control panel, this does not deal with the server resources directly but rather the types of object known by the Adapt Authoring Tool (e.g. users, projects/courses, file/data storage per tenant).
	Number of users	The number of users (per tenant) in case a service provider wishes to limit this (e.g. for number of seats type service provision).
	Number of projects / courses	The number of courses/projects (per tenant) in case a service provider wishes to limit this (e.g. for number of seats type service provision).
	Data storage	The amount of data on the file system and/or in the database in case a service provider wishes to limit this (e.g. first GB of data for free etc.)
	Backups	Setting to allow the automation of system and user data backups.
	Default values	Heading to help explore and organize default values. This can make a major difference in making the tool efficient for commercial content production purposes.
	Plugin architecture	Heading to help explore and organize a modular system structure in order to make it as simple as possible to extend and modify the system according to software development best practice. This heading also prompts everyone to think in terms of lowering the barrier to entry for the developer community.
	Package manager	User interface and program logic enabling non-technical system administrators to select and install extensions available (e.g. via contributions in the Github community) at the click of a button, even if they are in repositories owned by community members (as long as they have made the effort to 'register' the plugin). Examples of this are the Ubuntu Linux Synaptics Package Manager, Moodle's extension manager and any app store (to some degree).
	Metadata	Important data about the plug-in package. The metadata definition will vary between types of plugins and some metadata will be mandatory to enable important functionality (e.g. allowing the plug-in to fit into the authoring tool). For example, we would want to capture the following about a component plugin: data-schema (so that the authoring tool can render the data input fields), author/maintainer, platforms / browsers it works on etc.
	Languages	The language node captures two elements: The user interface language of the authoring tool as well as the language strings packaged with the content. We intend to make it easily possible to switch the language of the authoring

		environment as well as easily facilitate language translation for the learning content.
	Components	Components are plug-ins for the output (Adapt framework). Components are passive or interactive content elements, for example components are “Multiple Choice Question” or “Graphic”. These are developed as plug-ins in order for Adapt to be as flexible and extendible as possible.
	Extensions	Extensions are plug-ins for the output (Adapt framework) delivering additional functionality, but which are not components. For example: a glossary of terms.
	Workflow steps	Workflow steps are plug-ins for the authoring tool. As described below, we have split the key stages of working on a course as follows: plan, build, review, release, maintain, archive. The principle behind workflow step plugins is that we want to enable different organizations to facilitate different working practices.
	Themes	Themes are plug-ins, which define the look & feel of the authoring tool or the output (Learning Objects). Output themes define the overall look and feel of the Learning Object (output). This consists of base colours, background images and generic furniture, which will be used by components and extensions. Components and extensions can (should) inherit from the base theme and will have to implement their own layout separately.
	Menus	Menus are plug-ins for the Learning Objects (output), which provide the front page and structural navigation.
	MT (multi-tenancy) hierarchy	A multi tenancy hierarchy means that a single instance of the authoring tool will be usable by several sets of end users (tenants), who have administrative access and whose data is entirely separated and invisible to each other.
	Tenants	Tenants are set of users (usually organisations) who use the authoring tool independently of each other.
	Tagged folder structure	The purpose of the folder structure is to enable users to organize their courses / projects. By tagged folder structure, we mean a tag- or category-based filing system where individual courses can appear in multiple folders or categories. E.g. Google Mail approach rather than the more traditional ‘Windows explorer’ type folder structure.
	Courses (Projects)	A project or ‘unit of work’ in the authoring tool, which covers one contains a structure, content and is published together into a single output package.
	Owner	Owners are users with the highest level of permissions for a course. Ownership reflects how system roles and permissions are applied to courses. In Adapt Learning, we have chosen the owner/invite/collaborator pattern with collaboration and workflow in mind.

		This inspiration for this comes from online file sharing systems and systems such as Basecamp.
	Invites	An invite is an offer of access/editing/reviewing permissions from one authoring tool user (who has a rights, e.g. ownership, to assign permissions to another user). We believe an open, self-organising system is most appropriate for an authoring tool seeing as the tool is about creating an end result. This is reflected in the collaboration and workflow features. For reference, a contrasting approach where access permissions are tightly/centrally controlled is (often) required by LMSs, which are delivery / consumption focused systems.
	Collaborators	This term is used to reflect a number of users, who have access to a course, but lesser privileges (in terms of a system role) than an owner.
	Overrides	Each course can have overrides. Overrides are files (combined into a ZIP archive), which can be uploaded in to the authoring tool and will automatically be dropped into the overrides (or bespoke) folder of the published output. The course will use the override files instead of the core files as and when the filenames and folder structure match. They can also be used to package file resources (e.g. PDFs), which need to be contained within the course. This stems from experience of working with authoring tools and having to patch the output manually every time a course is published (e.g. to fix a bug or make a customization to the core output files). By providing override capability, it is possible to automate this process after the first execution. It also provides a way to modify core code without having to customize the tool or framework itself.
	Status, Statistics	Each course will have high-level usage data while being created and worked on. The status information captures this data and makes it available to any dashboards the tool may contain in time. Status information
	Dashboard	The dashboard is the view of high-level status information. The initial intended target audience is the Learning & Development manager system role. The idea behind this is that L&D managers may wish to view and report upon projects they are due to deliver (despite them not necessarily working on the actual projects themselves) as well as getting an overview of the usage of the system (which may be helpful in terms of buying decisions).
	Workflow	This is relevant for the authoring tool only. As described above under Workflow steps, we consider there is value in recognizing the workflow process and making this flexible and extendible, especially seeing as this is possible in a server-based system. With regards to the creation of courses, we recognize the following key workflow steps:
	Plan	The time during which the course is planned. This may include documenting overall design decisions, storyboarding, listing of learning outcomes. This may also be used to capture files / documents and use them for briefings, getting agreement or to refer back to during the later stages of a project.

		There is no specific functionality we have planned at this stage.
	Build	The time during which the course is built. This contains the majority of the expected content editing use cases.
	Review	The time during which the build is complete and going through any official period of QA, final changes and sign-off. Arguably, build and review (can) take place at the same time so there may not be a separation of which functionality is available during these two stages.
	Release	The time during which the course is officially released and launching. From a functionality perspective, this may simply be a view of the course history and he ability to take a snapshot to identify a version, which has been published and delivered. This may also be a connection point for 'publish to another system' type integrations.
	Maintain	The period of time during which the course is being kept up-to-date while after it has been published.
	Archive	The stage at which a course is compacted and potentially removed (exported) from the system. This may contain options about keeping or losing the editing history, ownership and other permissions etc.
	Users	The authoring tool operates on the principle of requiring system accounts for individual users. System users will be assigned system roles, which define what they can do in the system.
	User types	The authoring tool has a flexible, capability based system role component, which enables administrators to create roles from very granular actions. We do, however, recognize that there need to be pre-built system roles, which reflect the intended target audience of the system out of the box. The various user types below reflect this.
	Content Editor	This is the most common and functional role in the system and essentially describes a user who creates, edits and publishes courses.
	Guest user	Users without system accounts can be given access to the tool, e.g. for a review of a module. Guest accounts will be automatically created through invites (as part of the workflow) and can also be used by service providers to give access to potential interested parties to get a sense of the authoring tool and system. Guest users have limited viewing capability, the ability to comment and can generally not make any changes.
	L&D Manager	L&D Managers are high-level users. We consider they might be interested in an overview of projects, their status and the level of activity on the project. We assume that the most important use case for this user type is viewing overall system and per project dashboards.
	Demonstrator / Trainer	Demonstrators and trainers will use the system in a slightly different way. We would like to make sure these user types are covered to enable and establish service provision around the system.
	Developer	We expect that technical developers will use the tool very occasionally, e.g. to upload the overrides.

	Administrator	System administrators can execute all available actions within the scope of an individual tenant. Note, multi tenancy often entails there being a separate tenancy / super administrator account.
	User actions	Node to highlight some key actions to help express the vision for the tool.
	Preview	The ability to quickly view the course during editing without having to fully publish / download the course.
	Undo	The ability to undo previous action(s).
	Sign-off	The notion that system users can indicate their approval for the state of the course, e.g. via a checkbox.
	Versioning	The ability to capture the state of the course and its contents at a point in time. This could be used for releases or other workflow items.
	Publishing	The process of processing the data entered into a ready to download course according to the chosen output format (e.g. a SCORM module).
	Snapshots	Similar to versions (and maybe the same thing, unless we also capture revisions as version control systems do).
	Rollback	The ability to revert the course and its contents to a previous state.
	Output formats	Output formats are a plug-in for the authoring tool. Their purpose is to process the data entered as part of the authoring tool and convert it into the intended output package, e.g. a SCORM package.
	SCORM	Most commonly used standard for e-learning packages and usage tracking.
	Tin Can	Emerging standard for usage tracking.
	Review (action)	The activity of reviewing a course. *(See above for review workflow node)*
	Comments	The ability to leave a comment for a part of the course (e.g. an article, a component etc.), without making any changes to the content. Similar principle to comments in MS Word.
	Tracked changes Accept / reject	The ability to make changes to content and the system flagging these automatically and providing a simple workflow to accepting or rejecting changes. Similar principle to comments in MS Word.
	Import / export	The ability to import and export a course with the relevant data. This should enable courses being stored outside of the authoring tool (e.g. as backups) or exchanged between instances.
	Asset management	A part of the system, which enables the management of media files, including in particular imagery. The purpose behind this is to enable the management of reusable resources and be able to report on their usage. For example: • Reporting on the number of times a licensed image have been used • Ability to change an asset centrally and get an overview of which courses use it and would need to be re-published for the change to take effect.