Adapt Learning: Glossary of terms

Document control

Abstract:	Lists and explains the common terms used in the Adapt Learning Project to define a common language.		
Author:	Sven Laux	Version: 0.1	Date: 11 / 11 / 2013

Summary of Changes:	**Versions**	**Date**	**Description**
	0.1	11 / 10 / 2013	Initial draft for review.

Purpose of document

The purpose of this document is to list and explain the commonly used terms in order for us to establish a common language and avoid confusion.

Terms and explanations

ID	Node	Description
	Accept / reject (user action)	**Domain:** Adapt authoring tool The ability to incorporate or dismiss changes made directly to data entered in the authoring tool as part of a review.
	Accessibility	**Domain:** Adapt framework and authoring tool (published course) The principle of enabling users with a variety of impairments (in particular visual and motor type impairments) to use and navigate a published course (most important use case) as well as the authoring tool (lesser expected use case).
	Accordion (component)	**Domain:** Adapt framework and authoring tool An interactive presentation component, which is packaged with the core of Adapt. This is a single width component, which consists of a stack of clickable items. Each item has a heading which, once selected, expands to reveal the accompanying items text. There is no limit on the number of items within an accordion but we'd recommend no more than six. There are no graphics within each items display text area.
	Account expiry	**Domain:** Adapt authoring tool A policy, which defines the rules for the expiry user accounts in the authoring tool and the behavior of the tool regarding expired user accounts.
	Adapt Framework	**Domain:** Adapt open source project The Adapt Framework is the codebase, which runs as part of a published course as well as its developer interface. The Adapt authoring tool packages content and framework into a published course.
	Administrator (system role)	**Domain:** Adapt authoring tool A type of user account for the authoring tool, which can execute all possible user actions (has all capabilities) within the scope of a tenancy. Note, Administrator roles are limited to the scope of a single tenancy and are not able to handle the administration of tenants/tenancies. This set of capabilities is reserved for the 'Super Administrator' role.

	Agile methodology	**Domain:** Adapt Learning open source project The software development methodology we follow on the Adapt Learning open source project. More information at: http://en.wikipedia.org/wiki/Agile_software_development
	Application Program Interface / API	**Domain:** Adapt authoring tool Integration point for external systems, implemented as web services.
	Archive	**Domain:** Adapt authoring tool The term archive refers to multiple items: • The workflow stage of a project where a course is archived (and potentially exported) and is no longer live in the system. • The user action of archiving a course. • The physical set of files, which make up the course archive, with the relevant data from the authoring tool.
	Article	**Domain:** Adapt framework and authoring tool An article is a part of the course structure. It is the second biggest item in the hierarchy after 'page'. An article has a title and body (of initial text) and a background graphic and may contain specific functionality such as assessment.
	Assessment	**Domain:** Adapt framework and authoring tool An assessment is a sequence of questions, contained within an article, which can generate a score upon completion. Assessments have specific associated functionality and data, such as score, passmark, summary feedback etc.
	Asset	**Domain:** Adapt authoring tool and framework 'Assets' refers mainly to content images or animations used within the course. Assets are managed in the authoring tool and may be language dependent. Assets are packaged with the published course files and are displayed to the learner.
	Asset management	**Domain:** Adapt authoring tool A part of the system, which enables the management of media files, including content images in particular. Asset management functionality is intended to facilitate reuse and includes the ability to report on usage and make global changes, which apply to multiple courses.
	Audio/video preferences	**Domain:** Adapt framework Audio/video preferences enable learners to change the default behavior of audio / video assets for a published course. This includes enabling/disabling audio, setting the volume and deciding whether a written transcript for audio should be displayed alongside or instead of the media file.

	Authoring System, Authoring Tool	**Domain:** Adapt open source project A server based, e-learning authoring tool, with a graphical user interface, which produces responsive design e-learning content.
	Automation	**Domain:** Adapt framework and authoring tool The ability to run a number of otherwise manual developer tasks and executed them automatically on the basis of events in the authoring tool or commands invoked against the Adapt framework in the command line interface. We use Grunt (a JavaScript based Task runner) to automate tasks. More information at http://gruntjs.com/.
	Backbone	**Domain:** Adapt framework Backbone is a JavaScript based library, which structures our code and allows us to separate data from the display (by implementing the MV pattern). We also use its inbuilt event system to create event-driven program code.
	Backup	**Domain:** Adapt authoring tool A backup is a collection of data and files, which are used by the authoring tool to create a course. The authoring tool includes configuration settings (i.e. a backup policy) to determine the system's behavior. Setting to allow the automation of system and user data backups.
	Bespoke	**Domain:** Adapt framework and authoring tool A folder in the Adapt framework architecture, which can contain custom code not written as a plugin. This code may be uploaded as a ZIP file in the authoring tool and can be used to override core Framework files, thus enabling customisations without actually changing core code.
	Blank (component)	**Domain:** Adapt framework An empty, transparent component, usually used as a spacer allowing learners to see the background imagery.
	Block	**Domain:** Adapt framework Structural item with a title, body and background graphic. Placed into articles and contain components (two single width components or one spanned component). Can be used to implement alternative navigation through blockslider / blockview.
	Blockslider	**Domain:** Adapt framework A navigation method, which displays blocks in horizontal order and enables lateral navigation / scrolling (as opposed to deep scrolling).

	Bookmarking	**Domain:** Adapt framework The ability to store the last visited location in the course at the end of a session and returning the learner to this location upon starting the next session.
	Bower	**Domain:** Adapt framework A front-end package manager enabling developers to quickly download and install Adapt and plugins.
	Breakpoints	**Domain:** Adapt framework Screen / window width in number of pixels at which different styles are applied to the on-screen content to enable major layout changes as part of the responsive design.
	Build (workflow step)	**Domain:** Adapt authoring tool The project stage during which a course is built. This contains the majority of the expected content editing use cases.
	Bundle	**Domain:** Adapt framework Adapt framework core code including essential plug-ins.
	Capabilities	Individual user actions, which can be carried out within the system. This allows very granular control over system roles.
	Catalog	
	Codebase	
	Collaborator	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.
	Collection (class)	
	Command line interface	
	Comment	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.
	Compile	
	Completion	
	Component	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. Components sit within blocks and are used to present the course content. A component contains a title, a body (initial text) and a widget / piece of interactivity. This widget element is what differentiates the various component types. Any two single width components can be combined within a block. A spanned component will always have a single width

		full back to ensure it can be rendered on a smartphone-sized screen.
	Conditional loading	In the Adapt framework we conditionally load scripts based upon which browser the user is loading the course from. Modernizr comes with a built in yepNope conditional loader.
	Config / configuration	
	Content	
	Content Creator (system role)	
	Content Editor (system role)	This is the most common and functional role in the system and essentially describes a user who creates, edits and publishes courses.
	Core codebase	
	Course	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.
	Course structure	The arrangement of the various pages in a hierarchy, which makes up a course.
	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).
	Data separation	By having a clear distinction between our models and views we're able to structure and maintain a growing/large framework.
	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.)
	Deep scrolling	The most typical layouts are likely to be based around a deep scrolling page where blocks stacked one on top of the other.
	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.
	Demonstrator / Trainer (system role)	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.
	Dependencies	The RequireJS optimizer loads all of the Adapt dependencies in the correct order so we don't have dependency collisions or

		unnecessary script tags in the index.html file.
	Developer (system role)	We expect that technical developers will use the tool very occasionally, e.g. to upload the overrides.
	DOM	
	DOM manipulation	DOM manipulation allows us to move or affect DOM elements in the browser.
	Events	The Adapt framework is built upon an event system that enables our modules/classes to be separated and modular. One module should not talk to another module. Instead they trigger events that other modules can choose to listen to.
	Export	
	Extensions	Extensions are plug-ins for the output (Adapt framework) delivering additional functionality, but which are not components. For example: a glossary of terms. Extensions are a type of plug-in. The purpose of extensions is to enable developers to extend the functionality of the course without having to modify core code. Extensions contain functionality, which is not directly embedded in the article / block / component structure. For example, a course glossary and the 'Tutor' to deliver feedback.
	Feature detection	Modernizr attaches classes to the HTML tag based upon browser features. This can be used when styling or adding features and polyfills for different browsers.
	Feedback (extension)	Feedback is provided (via the 'tutor'; *see below*) for question components as either correct/partially correct/incorrect. Alternatively you can also present feedback specific to the option chosen.
	Fonts and symbols	Fonts and symbols are a consideration as special characters or particular fonts are not 'websafe' and may have to be shipped with course. This node is in the concept diagram to remind us of particular requirements when working with languages other than English.
	Glossary (extension)	Reference section showing terms used in the course and their descriptions.
	Graphic (component)	A single or spanned component. If spanned the same image will scale down and be 'redrawn' into a single sized component (Adapt doesn't load a smaller sized image).
	Graphical MCQ (component)	A spanned or single width component. Very similar to MCQ in terms of functionality, the only difference being the learner selects one or more icons on an image rather than one or more options from a list.

	Grunt	Grunt is a node based JavaScript task runner. In Adapt, this is used for automation.
	Guest user (system role)	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.
	Handlebars	Handlebars is our core templating engine. We precompile our templates through a Grunt process to enable fast load times.
	Hash tag	
	History	
	Hosting infrastructure	
	Hot Graphic	A spanned component containing an interactive graphic (an image with clickable regions whose coordinates are set via the JSON). When one of the items on the hot graphic is selected a window opens over the image containing some associated text and image that is relevant to the hot spot item selected. The learner has the option of closing this window and choosing another item on the hot graphic or using discrete forward and backward navigation buttons within the display window to work through all the display items in order (the order will depend on the JSON structure). For single spanned full-back, there is no mechanism for selecting items off the hot graphic. Instead the display window functions as a single span narrative.
	How-to guide (extension)	
	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.
	Inheritance (code level)	
	Instance	
	Integration	Multiple systems talking to each other directly, without manual intervention. Usually achieved by connecting systems via web service APIs.
	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.
	jQuery	jQuery is a cross-browser compatible DOM manipulation library. It enables us to write less code whilst working across all the currently supported browsers
	JSON	All of Adapt's data is stored in JSON format. JSON is formatted in a key and value pairing.
	Karma	We use Karma as a spec runner that runs our unit tests in a variety of browsers. This also integrates with Travis-ci and runs test through a headless browser known as PhathomJS.
	L&D Manager (system role)	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.
	Language	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. The language is one of the main configuration settings. Courses can be packaged to contain multiple languages within a single course package. The language setting determines the default language to be used and whether to display the option of choosing the language to the end learner.
	Learner support	Learner support is functionality that provides additional information, outside of the core content presented within the Adapt pages. Examples might be glossary, resources, page level progress and a search feature.
	Learning Designer / Instructional designer	
	Less	We use LESS as our CSS pre-processor. Less enables us to inherit and store variables that can be used across Adapt.
	Libraries	We use a variety of open source code libraries at the heart of Adapts core.
	Localisation / Translation	
	Locking	The ability to limit access to pages or specific blocks within a

		page until an event has occurred.
	Logging	
	Look & Feel	The appearance of the e-learning course, which includes branding, art direction themes etc.
	Maintain (workflow step)	The period of time during which the course is being kept up-to-date while after it has been published.
	Manager (class)	
	Matching (component)	A single width component that provides the learner with a stem presented via the component body, and then a series of statements. Each statement has an associated drop down which contains a series of options, one of which will be correct. All option drop downs require the correct option to be selected to trigger the correct feedback. There is also partially correct and incorrect feedback available. As with MCQ component the option marking, 'Reset' and 'Model answer' functionality is included. There is no limit to number of statements/drop downs or the number of options within the list but best practice would dictate no more than 5 or 6 for each.
	MCQ (component)	A single width multiple choice component. The body is used to display the question stem and the widget the options. One or more of these options can be marked as correct. Feedback, delivered by tutor feature, can be either option specific or banded as correct, partially or incorrect. Any 'number of attempts' are possible but recommend two as standard. The tutor window must be closed to reattempt the question. Indication of performance occurs via the placement of ticks and/or crosses. These icons are only located on the options the user has selected. If multiple attempts are possible, then the 'Submit' button is replaced by 'Reset' once the answer is submitted. Selecting this button removes markings and resets the question, allowing for a further attempt. Once the final attempt is made the reset button is replaced with the 'View Model Answer'. Selecting this button allows the learner to toggle between their own answer and the perfect answer. In model answer ticks and crosses are present on every option. There is no maximum number of question options but best practice would indicate a maximum of 5 or 6.
	Media file	Media files are also assets but refer more specifically to audio

		/ video files, which are used in the course. Media files are more likely to be language dependent, especially when they contain an audio track.
	Mediator (class)	
	Menu (extension)	Menus are plug-ins for the Learning Objects (output), which provide the front page and structural navigation. A mechanism for selecting a sub menu or page within a course.
	Metadata The process of navigating within and between the various pages of an Adapt course.	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.
	Minification	Running $ grunt build - a developer is able to minify all files including CSS and JS files. Minification means removing formatting, which is important for code-readability but not for machine processing, and thus making the code more efficient.
	Mocha	Mocha is a Behaviour Driven Development framework that enables us to run unit tests across our code.
	Model	
	Modernizr	Modernizr is a conditional loader and adds browser feature detection
	Modularity	Adapt is built upon a modular approach where modules shouldn't talk to each other and instead trigger events. This enables us to have a plugin architecture and smaller modular files.
	Module	
	Multi-tenancy / MT 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.
	Narrative (component)	A spanned component with a single width component fall-back. The learner can work through a sequence of images with an accompanying piece of display text via a forward icon (a back icon, post item one is also available). When the final item in the sequence is reached the default behaviour is to disable the forward arrow, not to take the learner back to the beginning of the sequence.

		Each display graphic and text field is accompanied by a title. The single spanned fall-back of narrative makes use of a clickable strapline which, when selected, triggers the display of the accompanying item text On mobile phones the display text is presented in a full screen pop-up, which needs to be closed before the learner can continue.
	Navigation	The process of navigating within and between the various pages of an Adapt course.
	Notifications	Notification is a messaging center that allows Adapt to flag information to the learner. In addition, the message can also provide choices which are linked to particular events (for example, on selection of the menu button 'There are still unfinished components on this page, are you sure you want to return to the menu. <YES> <NO>)
	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).
	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).
	Obfiscate	
	Output format	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.
	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. For developers we have a bespoke folder that enables us to do overrides / customisations of functionality without having

		to change the core code directly.
	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.
	Package	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.
	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).
	Page	A page is a structure, which consists of at least one article and a single block which houses one spanned or two single components. A page can contain as many articles as needed.
	Passkey	
	Passmark	
	Password policy	The authoring tool must have the ability to enforce rules that determine the strength of passwords and their expiry.
	Permission	
	Persistent nav-bar	The bar at the top of the page, which contains access to the sub menu feature (including 'Back' and the learner support features) and the page level progress.
	Plan (workflow stage)	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.
	Platform	

	Plug-in	
	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.
	Policy	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.
	Preferences	Preferences refer to settings the learner can change in order to adjust the display and behavior of the course. This includes audio/video and accessibility related preferences (such as high contrast look & feel) in particular.
	Pre-processing	CSS pre-processing enables developers to easily and quickly style courses, whilst inheriting styles, passing variables and nesting.
	Presentation types	Components, which focus on presenting information.
	Preview (user action)	The ability to quickly view the course during editing without having to fully publish / download the course.
	Progress (extension)	
	Project	
	Publish (user action)	The process of processing the data entered into a ready to download course according to the chosen output format (e.g. a SCORM module).
	QA	
	Questionbank	A bank is a collection of questions that can then be used, with randomisation, to create an element of variability between different sittings of the same assessment. There is, in theory at least, no limit to the number of banks or the number of questions within them. All question components can be used with a bank.

	Questions	Components, which focus on eliciting a measurable response from the learner. Questions can be used in formative and summative assessments and can contribute to an overall score for the e-learning course.
	Quota	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).
	Randomisation	Randomization is the process of, and rules governing, the selection of question from the question banks to create an assessment article.
	Redo (user action)	
	Register	
	Registry	We have our own plugin registry that enables developers to register their plugins (components, extensions, menus or themes). Once registered these plugins are available through the adapt-cli by running commands like $ adapt install adapt-contrib-text
	Release (workflow step)	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.
	Repository Adapt is built upon a modular approach where modules shouldn't talk to each other and instead trigger events. This enables us to have a plugin architecture and smaller modular files.	
	Requirement	
	Require JS	RequireJS is an AMD (asynchronous module definition) module loader. It enables us a greater sense of modularity and dependency loading.
	Resource	Resources are additional and related content items, such as reference materials and background reading. They may exist as part of the course or in downloadable format (e.g. as PDF documents).
	Responsive	Responsive design is at the heart of the Adapt Framework and enables single version output, which adapts according to the device resolution and capabilities. We have implemented responsiveness is by defining pixel-width breakpoints (*see*

		below). The Adapt Framework is fully responsive, meaning that as the window width increases or decreases, the content adjusts automatically. If the width (in pixels) goes beyond a breakpoint, different styles and classes are applied to the content.
	Rest	
	Restore (user action)	
	Review (workflow step)	Workflow step: 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. User action: The activity of reviewing a course. *(See above for review workflow node)*
	Roles	
	Rollback (user action)	The ability to revert the course and its contents to a previous state.
	Router (class)	
	RTL, Right-to-left (layout/formatting)	Depending on the language, the reading order may be right to left, e.g. in the case of Hebrew. This node is in the concept diagram to remind us of particular requirements when working with languages other than English.
	Schema	
	Scope	The major areas for which roles and capabilities apply. For example: • System wide • Tenant wide • Project / course wide Etc.
	Score	The score within an assessment is the number of questions answered correctly and displayed as a % or as x (correct answers) out of y (total number of questions).
	SCORM	Most commonly used standard for e-learning packages and usage tracking.
	Search	
	Security	A specific subset of configuration settings to make the authoring tool secure.
	Server	
	Settings	Settings reflect the configuration options of the generic codebase.
	Sign-off (user action)	The notion that system users can indicate their approval for

		the state of the course, e.g. via a checkbox.
	Snapshot	Similar to versions (and maybe the same thing, unless we also capture revisions as version control systems do).
	Specification	
	Statistics	
	Status	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
	Super administrator (system role)	
	System configuration	Settings, which determine the behavior of the authoring tool. 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 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.
	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.
	Tag/folder	
	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.
	Template	[…] We use Handlebars to load our views HTML. This enables Adapt to be dynamic and act as a one page web app.
	Technology Platform	The set of technologies (technology stack) that the Adapt Framework and the Adapt Authoring Tool are built upon.
	Tenant	Tenants are set of users (usually organisations) who use the authoring tool independently of each other.
	Tester / QA	
	Text (component)	A single or spanned component. This is the only 'proper' component (see blank) that doesn't have a widget consisting

		of a title and body. If spanned the same text will be 'redrawn' into a single sized component.
	Text input (component)	A single spanned component. As per matching, the question stem is presented via the component body but instead of statements with accompanying drop down lists, the learner is expected to enter their answers via a free text entry field. Each free text entry fields entered answer is checked against a string and if present then it's marked as correct. All free text entry fields need to be matched as correct to mark the component as correct. As with all questions, partial and incorrect feedback options are available. Number of attempts and number of question options are not limited other than by best practice. The option marking, reset, model answer and number of attempts are also standard and as described above.
	Theme (extension)	A theme is the generic look and feel. It captures display settings, which apply throughout the course. This includes CSS styles, icons, base colours and background images. Components and extensions inherit the base colours and generic icons but have to define their own layouts.
	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.
	Tin Can	Emerging standard for usage tracking.
	Topic	
	Tracked changes	
	Tracking	The ability to track a users progress through a course. This is done through SCORM or Tin-Can/xAPI and enables LMS's to gain information about the user during a course.
	Trainer (system role)	
	Transcript	A transcript is the content of the audio track (in particular) expressed as on-screen text. It enables delivery of the content to learners who are not able to play back or hear / understand the audio track.
	Triggered component	A component which is triggered via the selection of a link or icon that sits on the background graphic for the block, rather than one which is already displayed upon page load.

	Tutor (extension)	Tutor is the mechanism used to automatically present feedback upon the attempting of a question component.
	Underscore	Underscore is the utility belt for Backbone. It enables Backbone to manipulate objects like Backbone.Model and Backbone.Collection. Underscore also comes with some handy methods that help deal with arrays/objects.
	Undo (user action)	The ability to undo previous action(s).
	Unit test	Unit tests enable us to test single methods or functions in our code to make sure that produce the correct output. This is an important measure in order to achieve high quality code.
	User	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 actions	An action a system user can take in the Adapt authoring tool.
	User story	
	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.
	Utility belt	We use underscore when iterating over arrays or objects with _.each or with Backbone.Collection.each()
	Version	
	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.
	View	
	Wizard	
	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.
	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.