Fluid Content Aggregation (FCA)

Author: M. Lang Holloman
Position: Senior Systems Integrator
Computer Sciences Corp. - CSC
Navy eLearning Program - NeL
Pensacola, FL
Email: Michael.l.holloman.ctr@navy.mil

ABSTRACT

Group collaboration and individual contributions can now be realized through Fluid Content Aggregation (FCA). Current web-based presentation of content has progressed to being cross-accessible and fully integrated feeds/assets, encouraging collaboration among a range of business, organization, education, and user types. Advertising and News companies have recognized, that to promote, funnel, and control traffic, they must provide valuable up-to-the-minute substance targeted to the viewer, then provide the tools/resources to populate the content to other site locations. In addition, user input has become a standard feature of follow-up to excite users provisioning of additional analysis, commentary, and adding of content. This strategy has been realized, by pulling from resources and sharing embedded content among a diverse group of sites and resources.

Learners of today are now adapt to bank on this engaged form and are ready to contribute to a community's knowledge and education. Users would be provided permissions to add new package tree elements, re-orient the linear listings of elements, comments/discussions components provided by initialized element, and inline content input forms would provide the ability to rapidly include primary hosted content, 3rd party hosted content, feeds, objects, and formatting. Much of what is thought of as Web 2.0 would be exposed for use by course administrators, instructors, and learners through the input forms. Portability of the complete containment of all contributions and package contents would be exportable to allow organizations the ability to preserve and migrate the package among environments.

Industries of interest for this 'type' of content packaging and runtime components would range from universities, K-12, corporate, government, and DoD services. Content package typing would be extended, in this case, to classifying 'content type' to at least account for this type of 'collaborative', 'self-paced', 'instructor-led', etc. By providing a 'type' for the package, an entire business process would need to be further explored to account for all possible types and a list of components to by made available by the runtime.

Fluid Content Aggregation (FCA)

PROBLEM DEFINITION

Keeping content up-to-date or conclusive to the subject-matter is often accepted as the date of delivery. Meaning, if the content was developed 6 months ago, it could easily be out-of-date by now. How about funding to update the content? For a large corporation, university, k-12, or government services; the funding budget always weighs in, which will require a longer duration before the content is refreshed. Why can't content packages be fluid in nature depending on the contributions made to the package by the community of focus? Why can't one view or contribute to the activities or asset materials accumulated and sequenced by the focus community for the given content package? If a package had the ability to manage a class or project contributions, how would one archive a package to store all user contributions?

PROPOSED SOLUTION

Fluid Content Aggregation would comprise of the ability of a learner to contribute to the adding, updating, and deleting of items/activities/clusters in a content package. The course package would have the ability to be ever changing, depending on the amount of contributions made by the courses' audience/community. One would be able to use simple code editor text fields and/or WSIWG editors with common formatting, asset location snippets, page inclusions, etc. Course packages would be extended to include an overall 'Type'. The 'Type' would include possible labeling as 'Collaborative' to indicate that the runtime would display controls to a user to contribute content to the package. 'Collaborative' content packages would be considered a valuable collection of real course content, reference documents, pdfs, ppt, images, audio, video, flash, charts, text, web links, formatted text, feeds, etc.

USE CASES

Example 1 (Adding a Basic Hyperlink):
Learner 'A' would like to add a hyperlink reference to a paragraph of instruction in a musical history course. Learner 'A' would not edit the content provided by the original author, yet the launch able item would have the ability to be associated with additions of content. The original state of the content would not be authored. Instances of additions would be listed by title, contributor, and date created or last revised. In this case, Learner 'A' would select 'Add contribution', which would provide either a basic Text Field (HTML Editor) or the WSIWG & Text Field controls. The user would then input, in this case, a 'hyperlink' to the intended

destination using either the 'html' editor field or the WYSIWYG editor hyperlink prompt.

Example 2 (Add/Update Mechanical Maintenance Instructional Items): Learner 'B' works for a commercial repair facility, and is viewing a course on 'Small Engine maintenance for the Evinrude 9.9'. Learner 'B' notices that the instructions for the removal of fuel pump are indicated, but the installation instructions of the new fuel pump are not provided. Learner 'B' would then be able to 'Insert' a new launch able 'Item' or 'Activity' at the appropriate sequence point in the content package to now include "Installation of the Fuel Pump for the Evinrude 9.9 Engine 2000-2008 models". This item would then be linked to a pdf Learner 'B' had created with

Word. The storage would be either through a 'Media Repository' (media upload component presented at the time of the new 'Item' creation) or link to assets/media on another domain/host.

Example 3 (Managing & Measuring Projects):
Art Teachers 'A' & 'B' create a course package instance, which will have an instructional item of an html page that is comprised of various images and text relating to instructions/objectives for a project on 'Monochromatic Painting'. Students are then instructed to atleast upload a graphic file of their project painting, along with providing atleast (1) comment of another students painting. All of the graphic files will be located by individual student entry in the activity/items 'Student Paintings'. In addition, students are required to make at least (1) contribution to the 3rd Activity/Item 'Topic Contributions'. The students would then be able to provide files/links to perceived valuable content related to the topic of 'Monochromatic Painting'.

Now portability arises to afford the ability of the 'content package' to become exportable. All of the content would be exportable in the package based on the implemented specification standards required by the runtime in the package for local content storage. The links and includes of content would contain the same links and includes to the same host of those said files.

INTEGRATION AND TECHNICAL SPECIFICATIONS

1. Users will be provided with a mechanism to 'Add, Edit, or Delete(Archive) a main topic element provided by the business rule for permission of the assigned 'User type'. Example: a student would only be provided with the ability to 'Add' content elements, but could not 'Add/Edit' a main topic.

2. Content Package 'Type' to include the following: 'Collaborative', 'Self-paced', 'Instructor-led', 'Assessment', etc. The 'collaborative' type would dictate to the

runtime to present user controls based on 'User Type'.

3. 'Copy Snippet' code function to allow users the ability to 'Copy to Clipboard' the include and/or link code to reference the content currently loaded on screen.

4. Adding of main organizational or cluster elements would write the addition of the elements to the living 'fluid' manifest file. This is dependant on the specification which LETSI decides for SCORM 2.0.

5. Comments - Users will be provided with a scrolling text box of comments displayed according to each topic element which would be initialized. Users would also be provided with the mechanism to 'Post a Comment'. This action would write the comments to a static file associated in the package to the element initialized. This function would aid in allowing the package to be exportable to include all user comments.
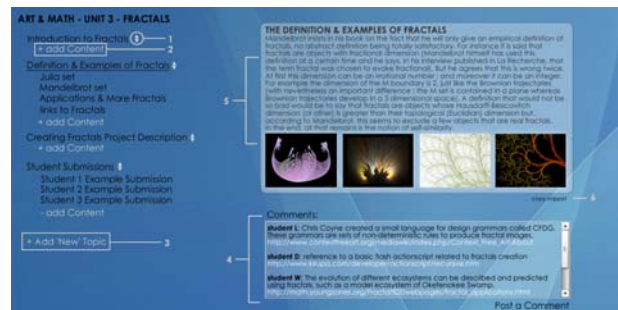
EXAMPLE IMPLEMENTATION
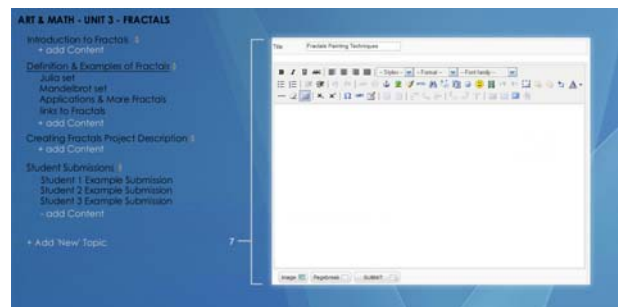

FIGURE (A) – FCA Content View

FIGURE (B) – FCA Input Form View

see last page of document for full graphic view Figure (A) & Figure (B)

Figure A.1 – Represents a basic control to raise/lower the element item up/down the content tree.

Figure A.2 – Represents the (+) for adding a new nested item.

Figure A.3 – Represents the (+) for adding a new topic item.

Figure A.4 – Represents the comments read section for the particular item, which is currently launched. The user engaged also has the ability to post a comment.

Figure A.5 – Represents the launch able content item window or 'main content window'.

Figure A.6 – Represents the content link, which provides the user the ability to attain the include code for the current assets displayed in the 'main content window'. This is one of the most important components that can make or break the ease of use of users to easily include this content in other package aggregation locations. The world all knows how to include images and text to other sites in everyone's 'comments' on social networking sites. What does every online media company provide for their content that is 'shareable'?...They provide the code to include their content on other sites.

Figure B.7 – Represents the WSIWYG editor for adding and updating content. A full set must be required and all of the W3C standards for css and html must remain in tact. The user will also have the option of running in 'open-code' editing mode.

*Not displayed in these examples are design / user-interface components typically contained in content. The User Interface concepts should be treated as a separate topic for analysis and implementation. UI and FCA do have overlapping functional aspects, but also need to be addressed separately.

SUMMARY

Fluid Content Aggregation (FCA) provides the mechanism and interactions to promote community contributions adding/updating content package data. Inline content input forms would provide the ability to rapidly include primary party hosted content, 3rd party hosted content, comments, and formatting. Lastly, the portability of the complete containment of all contributions would be exportable.

FIGURE (A) – FCA Content View

FIGURE (B) – FCA Input Form View