Designing and implementing a user-generated content service for Europeana

Nicola Aloia, Cesare Concordia, Carlo Meghini.

Istituto di Scienza e Tecnologie dell'Informazione, ISTI-CNR, Pisa, Italy,
(nicola.aloia,cesare.concordia,carlo.meghini)@isti.cnr.it

Abstract— **The paper presents an overview of the user generated content service that the ASSETS Best Practice Network has designed, implemented and evaluated with the user for Europeana, the European digital library. The service will allow Europeana users to contribute to the contents of the digital library in several different ways, such as uploading simple media objects along with their descriptions, annotating existing objects, or enriching existing descriptions. The user and the system requirements are outlined first, and used to derive the basic principles underlying the service. A conceptual model of the entities required for the realization of the service and a general sketch of the system architecture are also given, and used to illustrate the basic workflow of some important operations.**

Keywords-component; user-generated content; digital library; Europeana; UGC; EDM; ESE.

I. INTRODUCTION

In the 2011-2015 Strategic Plan, Europeana [1] announces User Engagement to be one of the strategic tracks by which the organization will deliver value. By the term 'Engage' Europeana refers to cultivating new ways for end user to participate in their cultural heritage. The Europeana network comprises communities of archivists, curators and librarians who show a growing interest in exploring new methods of access and dialogue. Europeana intends to enhance the user experience and offer services that allow users to interact and participate.

User-generated-Content (UGC) is one aspect of this renewed way of participating. Information about cultural heritage exists outside the heritage institutions; artifacts, written sources and memories of individuals complement collections held in institutions. UGC services are designed to provide users with means to support and interpret content. They will be involved in storytelling, curating of virtual exhibitions, reviews and even the creation of new collections. Greater participation will increase users' interest and loyalty. Europeana is therefore devoting increasing resources to initiatives that bring out the value of the contribution those users can make.

In response to these needs, the ASSETS [2] Consortium has included the support of user-generated content amongst the services it is going to develop for Europeana. ASSETS is a two-year Best Practice Network co-funded by the CIP PSP Programme to improve the accessibility and usability of Europeana. The ASSETS Consortium comprises 24 partners, including institutions from ten different European countries and Japan, which are active in the field of cultural heritage and digital libraries.

Rather than focusing on a specific set of UGC applications, ASSETS developed a general purpose, back end component that aims at supporting any UGC service Europeana will want to offer to its users. To this end, the ASSETS back end component implements an Application Programming Interface (API) for creating, storing and manipulating UGC Units of Work, and for submitting these Units of Work to Europeana, in the form of Europeana Submission Information Packages (SIPs). Final users will interact with their Units of Work through client interfaces, which will hide the unnecessary technical details and complexities of the back end to them, providing them with the level of representation that is most suitable for the specific UGC task at hand. Indeed, it is expected that every UGC task will be supported by a different final user interface. But this will have no impact on Europeana, since every different front end will talk to Europeana through the same API.

The API relieve future UGC applications from implementing any server side functionality; they will have to code only the client side, connecting the user world to the API. At the same time, the API move away from Europeana the technical interoperability problems that would arise upon integrating into its database the possibly different objects coming from future UGC applications. The service rely on the Europeana Data Model (being developed by the Europeana version 1.0 project [3]) in order to tackle the more serious semantic interoperability problems.

The definition of the conceptual model underlying the UGC API is the most difficult challenge that the ASSETS UGC team has faced. The proposed model attempts to provide the optimal balance between simplicity, so to be quickly learned and easily coded against by the future UGC service developers, and generality, so to satisfy the needs of any possible future UGC service. This conceptual model has been defined during the first year of the ASSETS project, based on an analysis of the different types of requirements that are in place. The model has been subsequently used to define the UGC API, which has been implemented and demonstrated at the first year project review, in June 2011. On top of this API, a few, initial UGC tasks will be implemented through specific front ends. These tasks will allow Europeana users to contribute to the contents of the digital library in several different ways, such as uploading new objects along with simple descriptions, annotating existing objects, or enriching existing descriptions.

The paper presents an overview of the UCG service. The user and the system requirements are outlined first, and used to derive the basic principles underlying the service. The conceptual model required for the definition of the API and a general sketch of the system architecture are also given, and used to illustrate the basic workflow of some important operations.

II. REQUIREMENTS

A. User Requirements

User requirements can take different forms:

- First of all, the submission of objects to a repository. Through a user interface (typically a website), users upload digital objects of various formats (images, audio and video). These must be objects that they actually hold rights over (for instance, by creating them). A minimal set of metadata will have to be provided in order to support the interpretation, discovery and management of the object. The user requires to be free of choosing which metadata format to use, but the system must propose a default one.
- Secondly, metadata enrichment. Users contribute factual metadata to an object, such as location, date, names, or tags. The object can be created by the user, but also by another user; the object may also enrich existing content in Europeana.
- Thirdly, annotations: users are contributing their views, comments. opinions to an object
- And finally, there is contextualization: through storytelling or creating virtual exhibitions and galleries and possibly adding narratives to them, users are combining existing objects into a new context (without changing the objects and metadata itself).

Before publishing user generated content, moderation may be added as an intermediate step in the process. Authorized users review the UGC and decide to accept and publish it. In most cases, this includes a feedback loop to the user who originally contributed the data.

B. User Interfaces

Europeana audiences include academic researchers with a high level of language and computer skills but also people who are hardly familiar with foreign languages or using the internet. Typical target groups are secondary school students and their teachers as well as cultural explorers and travelers. While possessing intermediate to good knowledge of foreign languages and online search, these groups generally expect services to be easy and intuitive. At the same time, they want to understand what happens with their contribution and who keeps control over their content. Secondary school students are generally known for focusing on images rather than digesting long texts.

User Interfaces should therefore preferably be simple, straightforward and visual. Additionally, clear information must be provided about rights regarding the content.

C. System Requirements

The back-end component has to comply with the Europeana architecture, which is based on an Open Source policy. Europeana has also defined a set of guidelines regarding the coding, the testing and the deployment of the components that make up its architecture. These guidelines are publicly accessible on the wiki of the EuropeanaLabs [4] and are been followed by the ASSETS UGC development team.

One important requirement concerns the management of multimedia content. As outlined in the previous section, Europeana expects users to contribute media files of various formats including text, images, videos and the like through its UGC service. However, Europeana was not initially meant to deal with (i.e., collect, store and make accessible) media files. With UGC, this has to change, of course. Moreover, the change has to happen gradually, by evolving the current architecture in a stepwise manner, so as not to compromise the operation of the existing components. In order to meet this requirement, ASSETS provides a simple media object repository, and deploy it on a separate server, the ASSETS server, to be later integrated into the Europeana architecture. More details on the media object repository are given in the architecture Section.

III. THE CONCEPTUAL MODEL OF THE UGC SERVICE

In order to meet the user and system requirements, the ASSETS team designed a UGC service based on the concepts outlined below and presented in Fig.1 as a UML class diagram. In the diagram, boxes represent classes and arcs represent associations. Every association is bi-directional, but only the name in one direction is given for readability purposes. The name given refers to the association going from the class closest to the name to the other class. The reader can derive the name of the association in the opposite direction by following any notational convention.

From the UGC server point of view, at any point in time there exists a set of users of the UGC service. Each user is in fact a role, identified by an id and a password, behind which a whole community may actually operate. Each user has its own Workspace (WS) on the UGC server. A WS is simply a container of the objects that the associated user needs to perform UGC tasks.

The creation of a single UGC object may take a long time and span several sessions of work. In between one of these sessions and the next, the partial results achieved so far have to be persisted, in order not to be lost and to be resumed at the beginning of the next session. The concept of "partial" UGC is captured by the notion of Unit of Work (UoW). The UoWs of a user are maintained in the user's WS.

A single UoW contains objects, identified by URIs, and their accompanying descriptions. The objects in a UoW can be of two kinds:

- Existing Europeana objects, that the user has included in the UoW in order to link them to new objects (see below) as values of some property, or in order to enrich them with new descriptions. Existing Europeana objects can be retrieved for inclusion in a UoW via a query issued to Europeana.
- Newly created objects, which are called UGC objects. These objects are original contributions to Europeana, and can be of three kinds:
    - i. digital objects having an associated media file with the content;
    - ii. digital objects for which no media file is available;
    - iii. non-digital objects.

consisting of a property and a value. Different attributions can have the same property with a different value. A value can be itself an object, or a literal or another resource, external to the digital library.

When a UoW is ready to be submitted to Europeana, the user can do so by using an operation that transforms the UoW into a well-formed Submission Information Package (SIP) and places a message signalling the existence of the SIP into the Outbox. Each user WS is endowed with an Outbox. Europeana retrieves messages from Outboxes in order to harvest the corresponding SIPs.

As already mentioned, users can issue queries to Europeana in order to retrieve objects. Each query returns a result, in the form of a message stored in a special area of the user WS called the Inbox. For each retrieved object, the result of a query contains a subset of metadata of the ESE schema of the object, a URI to the full set of metadata and a URI to the digital object on the content provider's web site, if any.

Each user WS is endowed with an Inbox. Messages in the Inbox are of two kinds: query results and notifications that communicate the result of submissions. These latter notifications may be positive and report the successful ingestion of a SIP, or may be negative and report the reasons why a certain SIP could not be

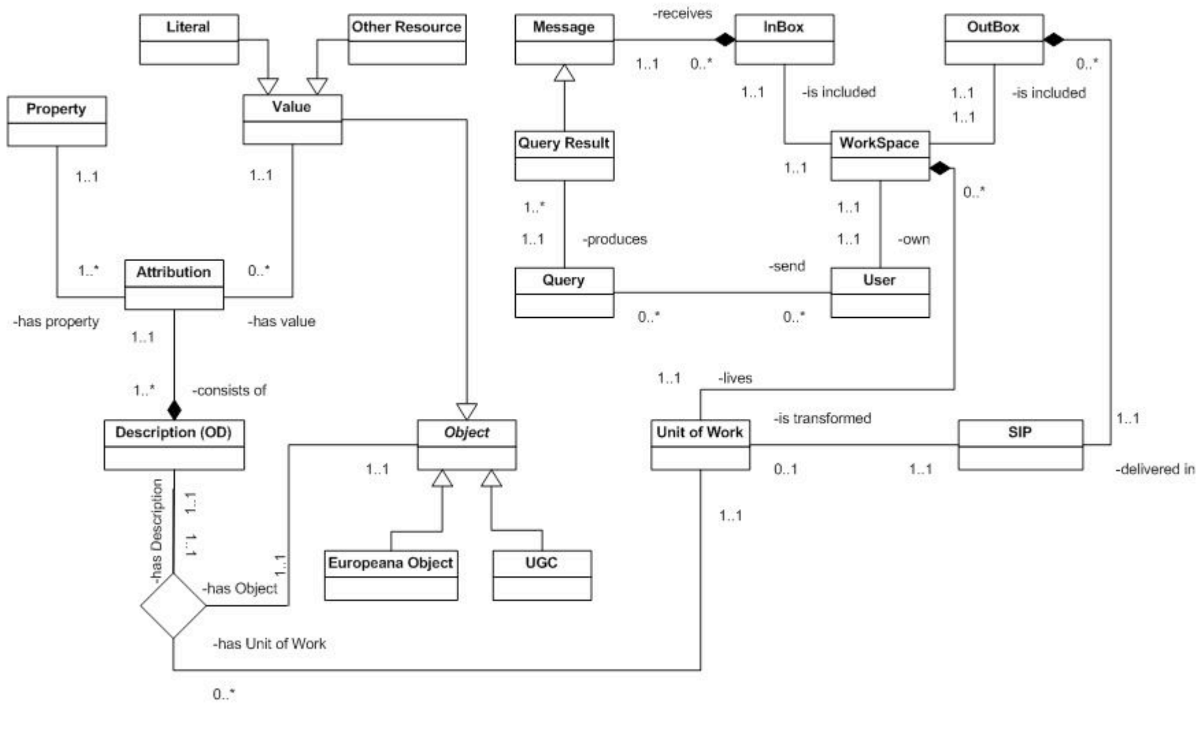

Figure 1 UML class diagram of the UGC service conceptual model

Every object in a UoW has an associated description. A description represents a metadata record of the object and is modeled as a set of attributions, each attribution ingested. In the latter case, the rejected SIP can be retrieved and re-transformed into a UoW so to allow the user to perform the necessary repairing actions.

It is important to notice that these concepts define a general-purpose schema, whose machinery need not be used by every UGC application. For instance, a simple UGC task that takes place in a single session, such as an image upload, may be implemented by directly building the corresponding SIP, so by-passing the UoW stage. On the other hand, another UGC application may decide to publish a finished UoW to a community of users in order to perform a socially oriented form of mediation before submitting the UoW to Europeana. These decisions will be taken by the client side of the applications, relying on appropriate shortcuts offered by the UGC API.

IV. Architecture

For the purposes of developing, testing and evaluating with users the UGC functionality, the UGC server will be deployed on the ASSETS Server. After successful evaluation, the Server will be moved into the Europeana production server and its functionality will be made available to the Europeana users. Fig. 2 shows the architecture of the UGC service. The three main components of this architecture are:

1. *Europeana Server.* This is the server implementing the Europeana search functionality. It provides an API to search for content in the digital library, using optional filters on metadata fields. Europeana query API implements the Open Search directives (http://www.opensearch.org/Home). The Europeana server also provides an application for harvesting of data (e.g. Repox, is a GUI based tool for managing these activities).

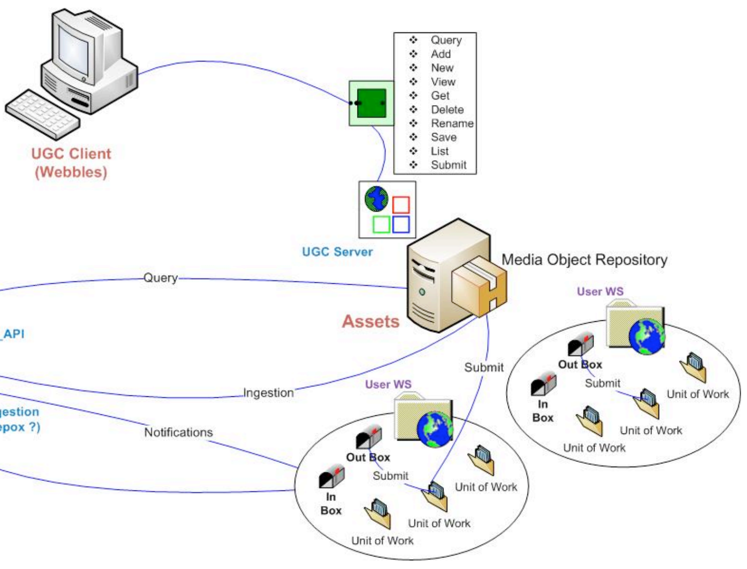

Figure 2 UGC service architecture

2. *ASSETS UGC Server.* In the context of UGC, the ASSETS UGC server provides functionality to communicate with Europeana and with the UGC client. The communication with the Europeana server is by invoking the Query API module (namely OpenSearch API). The ASSETS UGC server manages the workspaces of the users, and provides an API to manipulate the UoWs in the workspace. The UGC server provides API as REST Web services and is independent from any specific UGC client. The User WS contains an Inbox, an Outbox and the set of Units of Work that the user is currently playing with. In addition, the UGC Server maintains the ASSETS Media Object Repository (AMOR), logically partitioned into two sub-repositories: the repository storing the SIPs not yet harvested by Europeana, and the repository storing the SIPs already harvested by Europeana but not yet ingested by it. On the former repository, AMOR implements OAI-PMH functionality to allow Europeana to harvest SIPs.
3. *UGC Client.* Is a browser-based GUI supporting the user in a specific content generation tasks. The UGC Client interacts with the ASSETS Server via REST web services provided by the UGC Server module

V. The ASSETS UGC Demontrators

To verify the functionality of the UGC services, two demonstrators have been implemented. The first one is a simple HTML page with examples of use of the API provided by the UGC service. This demo is useful for developers who want to implement GUI using HTML and is available at [5]. The second consists of a GUI that uses the "webble" technology developed at the MEME Media Lab and is available at [6]. In Fig. 3 a screenshot of the last demonstrator is shown. On the left side of the GUI there is a tabbed pan having the following sections:

- *Uow* -- Shows the list of existing UOWs (possibly empty) on the Assets server. The user can create a new UOW by dragging objects from "Europeana search" and/or from "User Object" Tab.
- *Search Europeana* -- The user can search Europeana objects by querying in Google style (also an advanced search using Boolean operators among formatted fields condition is supported by the backend). The user will get a set of objects that can be dragged in the yellow area of the GUI, to create UOW.
- *User Object* -- The user can upload an object from his workstation, or can get an object from the web (by providing the object URI). The user can drag these objects in the yellow area of the GUI, to create a UOW.
- *Property* -- The user will see on the left side the list of the ESE properties that he can assign to the objects that are in the yellow area of the GUI. Complex UGC object can be created by using ESE properties like "Has part"/"Is partOf"/"Is VersionOf/etc.

VI. Conclusions and Outlooks

The main concepts and architectural features of the user-generated content service have been illustrated. The service has been implemented by the ASSETS Best Practice Network and will be evaluated within the lifetime of the project.

The UGC service developed by ASSETS is based on a general-purpose back end, which is meant to relieve Europeana from dealing with the specificities of the possibly very many UGC tasks that may be offered to users. At the same time, the back end relieves developers of UGC tasks from implementing the server side of their applications.

Acknowledgment

This work has been partially supported by the PSP Best Practice Network ASSETS (CIP-ICT PSP-2009-3, Grant Agreement n. 250527).

References.

[1] www.europeana.eu.
[2] www.assets4europeana.eu
[3] version1.europeana.eu/web/europeana-project/
[4] europeanalabs.eu/wiki/.
[5] http://146.48.83.12:8983/assets/communityservices-ugc/.
[6] http://www.meme.hokudai.ac.jp/WebbleWorld/WebbleWorldIndex.html.

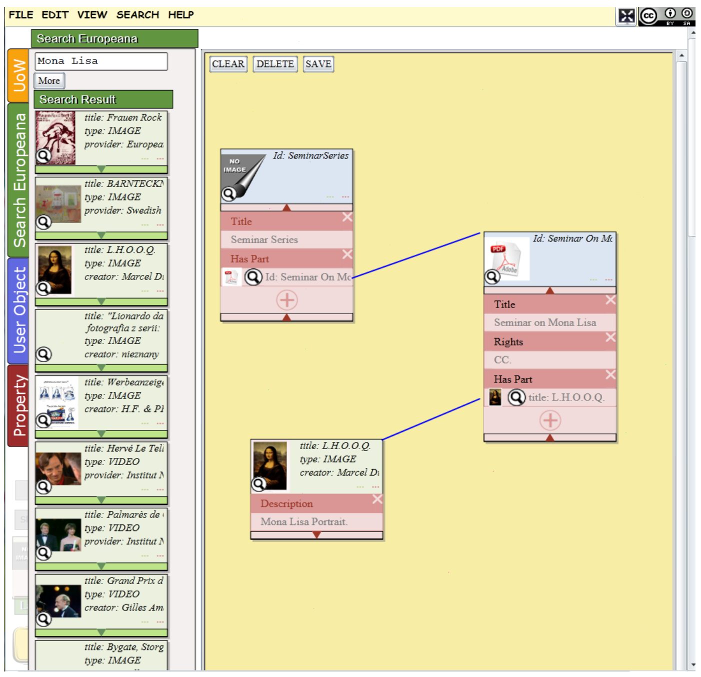

Figure 3 The GUI demonstrator based on webble technology