Roadmap Production Process

DDI 4 Sprint in Toronto, June 2014

Introduction

We started with the concept for a production workflow as designed in the previous sprint in Vancouver (figure 1) and identified a couple of issues (documented in the Wiki: IASSIST sprint, technical group). Because we were able to solve most of these issues during the sprint, this document focuses on only a subset of issues where the solution is important to document for subsequent work.

Out of our 11 issues, we selected one as our overall goal for this sprint: to produce a working prototype for the production workflow to prove the overall concept.

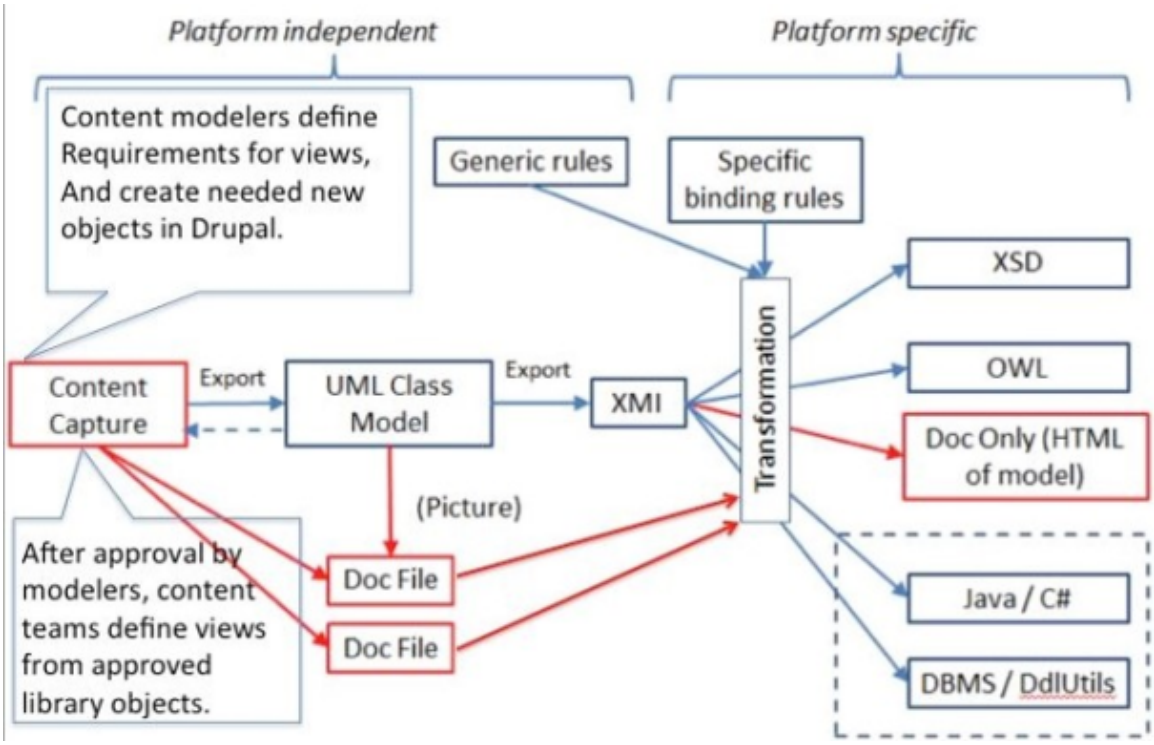

Figure 1: Production system as designed in Vancouver (deprecated)

Issues discussed

Identifiers

We decide to use UUIDs to identify objects and packages (incl. views). We do not want to use the object names because they are not stable during development and we also do not want to use the internal identifiers from Drupal because the might break with modifications of the technical infrastructure.

Joachim identified a method to roundtrip IDs from Drupal. This is an important preparatory step to actually shipping IDs forth and back. XMI allows the integration of custom attributes attached to any type of object

```
<ownedAttribute visibility="private" name="CustomLabel2" xmi:type="uml:Property"
xmi:id="EAID_084B6228_A680_47fa_8985_F729C4CAD9D6" isDerivedUnion="false"
isUnique="true" isOrdered="false" isDerived="false" isReadOnly="false" isStatic="false">
<type xmi:idref="EAID_7DF1F728_66C0_4376_A7EE_651E86DF2986"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000015_A680_47fa_8985_F729C4CAD9D6" value="1"/>
<upperValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000016_A680_47fa_8985_F729C4CAD9D6" value="1"/>
<defaultValue xmi:type="uml:LiteralString" xmi:id="EAID_LI000017_A680_47fa_8985_F729C4CAD9D6"
value="QuestionnaireLabel2"/>
</ownedAttribute>
```

XMI example with embedded identifier from Drupal for the round-trip with EA.

XMI export from Drupal should put its own UUID there. The information is confirmed by manual testing to be both displayed in EA, exported safely and visible again on re-import. Behavior on object changes, deletion, etc. needs to be determined.

Update:

UUIDs are now automatically created in Drupal for all packages and objects. These get exported in the xmi export from Drupal. However a prefix (ddi4-)needs to be added as xml id doesn't support numbers as first character. will be used as prefix.

> *Example: ddi4-0bea67ee-c3e0-4620-b8f0-3f763d734ec8*

In the docbook export from Drupal a suffix might also be added to specify sections in the document.

> *Example: ddi4-0bea67ee-c3e0-4620-b8f0-3f763d734ec8-definition*

Diagrams

We identified a basic structure for diagrams to document our model.

1. The model of the entire library
2. Show the packages in the library and their relations
3. Show the objects in each package
4. Show each object and its directly related objects (like clickable GSIM)
5. For each view
6. A diagram of all the objects in the view
7. A diagram for each object (same as 1. c)

Formats other than XSD and OWL

Formats other than XSD and OWL will not be part of the official standard specification. Nevertheless, we like to provide basic support for other formats (e.g. SQL, Java, C#, JSON). There will be a discussion within the DDI developers group to prioritize possible formats.

Some work on the combined use of SQL and Java has already been done.

Single source of information

From a long-term perspective, it would be desirable to have a single source of information for all objects and packages but also the surrounding material like the high-level documentation. An interim solution might be to store the XMI and DocBook files in a Git repository. Further solutions are discussed later in this document.

Updated image for the production workflow

After substantial discussion of the production workflow (Vancouver version) we did some modifications to the design, documented in figure 2.

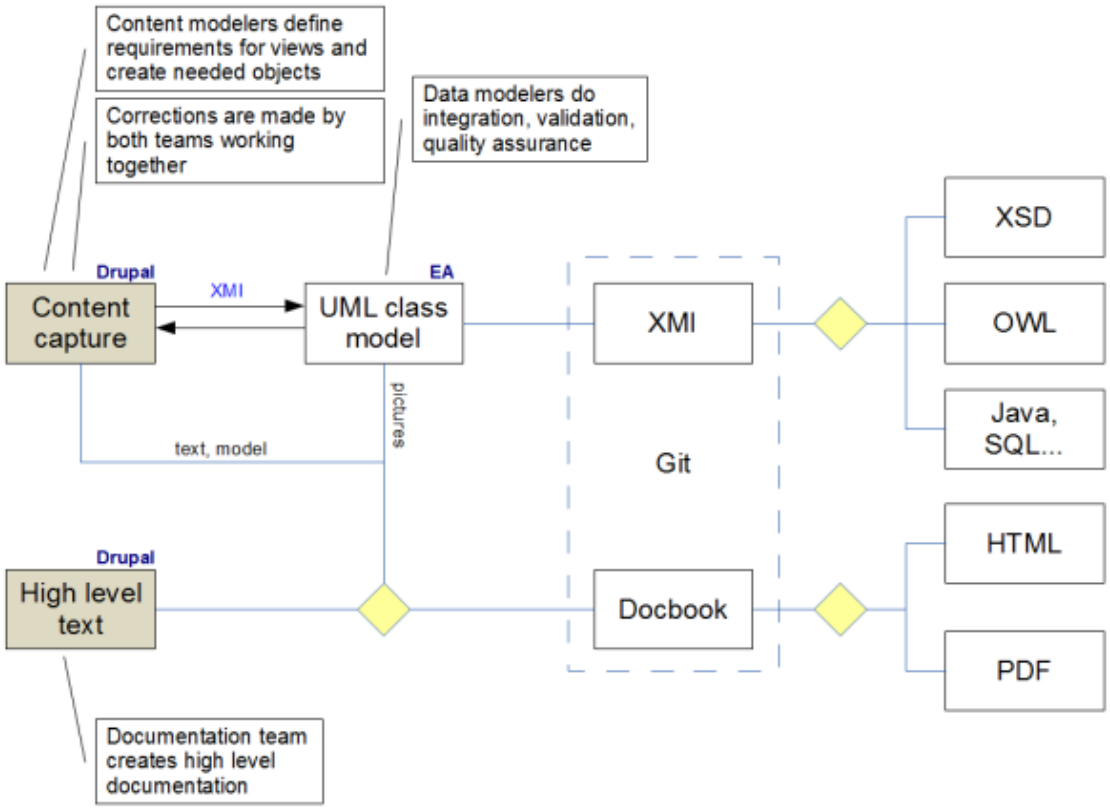

Figure 2: Production system as designed in Toronto

Handling updates in Drupal and EA

The most crucial problem is the fact we might have significant changes to the model in two different software environments – Drupal and EA. To ensure a coherent model we discussed three options to prevent us from collisions. First, all changes happen in Drupal and EA is used only for the generation of graphics and as a validation tool. Second, we implement a round-trip between Drupal in EA that would allow us to move objects between packages in EA and update Drupal using EA's XMI export. Third, one central repository (e.g. Subversion or Git) is implemented and both, Drupal and EA, interact with it. These three options are discussed in the following.

Option 1: All changes are done in Drupal

Drupal is the authority on the modelling. It contains all the information on the model, including relationships, objects and documentation. Technically, everyone can make changes at any time, which can be seen by anyone in real time. The feedback process between content modelers and data modelers is facilitated through Drupal. You can use other documents, but it is not real, if it is not in the Drupal.
There is an export to Enterprise Architect (EA), so diagrams can be produced and additional annotations for the XMI can be added (if needed).
The workflow diagram is identical to the one in the slides, just that there is no feedback between Drupal and EA (hence the one-arrow option).
Advantages:

- Technical infrastructure already exists and is proven to work
- Constant access to the information source for everyone
- Modelers are not required to use EA (own a copy, etc.)
- Non-modelers are encouraged to enter data, e.g. on documentation

Disadvantage:

- Modelling work that has been done in EA is going to have to be fed manually into the Drupal system (this can be source of error)
- It can also be tiresome for some tasks, such as re-packaging
- specialized scripts for Drupal may be implemented to increase speed for these tasks

Option 2: Round-trip between EA and Drupal

The Drupal DDI4 installation will be able to import an XMI export from Enterprise Architect -EA and update the content information residing in Drupal.
The round trip should be able to create / update / delete the following elements:

- package
- class
- properties of classes
- association

The most important requirement is to move classes between packages. Text documentation (e.g. description) will not be included in the round-trip. In the XMI a custom identifier is introduced and populated with a Drupal specific identifier. This will enable Drupal to relocate created elements.
Change and versioning:

- Only delta changes from EA to Drupal go back into Drupal.
- Establishment of an overall global checkpoint -OGC of the whole model.
- An OGC is an aggregation of all packages and classes at a given point in time defined by the specific versions of all packages and classes at that specific moment.
- The OGC feature can be used to 'checkout' a previous OGC version out from Drupal and a potentially import into Drupal would rollback an EA import.

As an import from EA has the potential to result in a lot of changes to the model in Drupal a restriction is set in place to restrict this model batch change possibility.

- Only modelers will have the credentials to import into Drupal

As we will not develop a versioning control system for modeling content but aim to avoid merge conflicts done by modelers. The shared folder and the collaborative Word doc editing will be implemented.

- When a modeler exports the XMI from Drupal the modeler XMI export is locked to that modeler.
- The modeler XMI export lock is released when the XMI is either reimported into Drupal or that the 'checkout' modeler releases the lock without importing.

Technical estimates to be defined in order to fully prioritize EA to Drupal import:

- Identify delta change from EA to Drupal
- OGC definition and display in current Drupal setup
- OGC XMI export
- Modeler XMI import credentials
- Modeler XMI export lock and notification options to other modelers

Option 3: Repository-based infrastructure

Drupal and Enterprise Architect could have a common repository for storing the information on the packages and objects.

EA has the possibility to use a versioning system (i.e. Subversion) for version control.

Citation from "Version Control Best Practices for Enterprise Architect":

> *"In this scenario we have multiple individuals editing the model, but without being connected via a shared Model Repository. Instead, each editor maintains a local copy of the model as an EAP file and periodically updates their copy from a shared Version Control Repository. This approach facilitates broad-scale replication of the model without the need for administering a DBMS."*

EA can version per package. XMI is used as transport format between EA and version control repository.

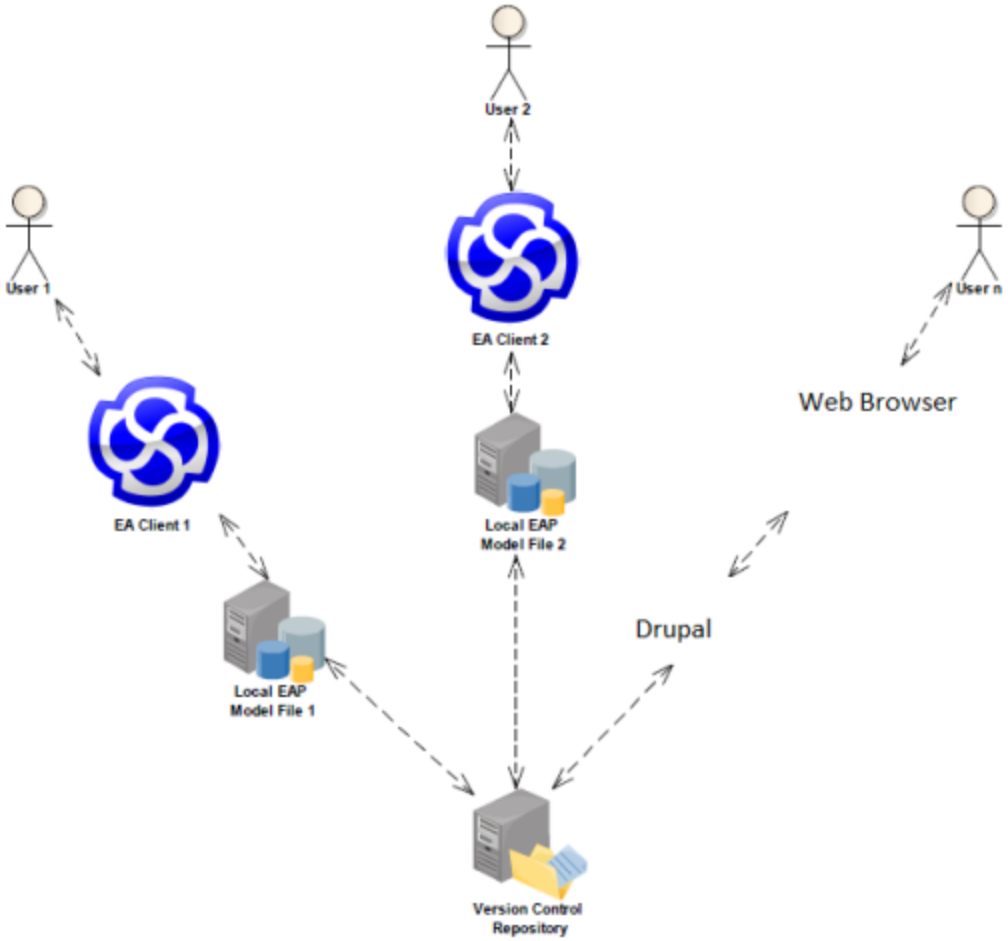

Figure 3: Using Drupal and EA with one central repository

Drupal could use the version control repository as source for the own data base tables.

The database scheme would need a structure according to the items used in the XMI file, possibly a table for each the packages, the objects, etc.

Advantages:

- single repository, no synchronization between Drupal and EA necessary
- common repository for multiple remote EA users

Changes of Drupal are required:

- possibly in the database structure
- regarding usage of identifiers to reflect the XMI structure which is used by EA
- (Basically, this would also make sense for the synchronization of the Drupal and EA by other mean.)

Possible issues:

- creation of a new package/object in Drupal should generate the same XMI structure as EA would generate

Possible business rules:

- only one person should work on one package at the same time
- update of Drupal should be started manually after update of package in Drupal
- While repacking in EA, no work is allowed in any package.

Discussion

After some deliberation, we decided to start with option 1 and make all changes in Drupal only. This would help us to start with a first prototype soon but would still allow as to additionally implement one or both of the other options. From a long-term perspective one single repository (option 3) would be desirable. Some tasks might be easier to archive in EA, so option 2 will be subject to further evaluation. At the moment we might not

have enough development resources for option 2 and 3.