MIGRATION BETWEEN CONTENT MANAGEMENT SYSTEMS
Exploiting the JSR-170 Content Repository Standard

Michael Nebeling
Monash University, Australia/Ulm University, Germany

Grace Rumantir, Lindsay Smith
Berwick School of Information Technology, Monash University, Australia

Keywords: Enterprise Content Management (ECM), Content Management Systems (CMS), CMS Migration & Integration, CMS Interoperability.

Abstract: Content management systems (CMS) have evolved in diverse ways due to the lack of sufficient standards. Even amongst CMS supporting JSR-170, the recent Java standard to organise and access content repositories, incompatible content structures exist due to substantial differences in the implemented content models. This can be of primary concern when migration between CMS is required.

This paper proposes a framework to enable automated migrations between CMS in the absence of consistency and uniformity in content structures. This framework serves as a starting point of a body of research to design a common content model which can be implemented in future CMS to facilitate migration between CMS based on JSR-170 and improve integration into existing information systems environments. This is illustrated via a simple website created using two of the most popular open-source CMS supporting JSR-170: Magnolia, and Alfresco. A model-based approach towards a generalised content structure is postulated to resolve the differences between the proprietary content structures as identified in the visualisation of the simple website created. The proposed model has been implemented in Jackrabbit, the JSR-170 reference implementation, and the proposed framework therefore contains simple methods to transform content structures between Magnolia and Alfresco using this Jackrabbit implementation as an intermediator.

1 INTRODUCTION

Content management systems (CMS) provide the set of technologies to support the processes involved in the management of content lifecycles with software. In a web context, common processes include creating, updating, publishing, and archiving web pages that may contain text, images, and other forms of multimedia. For CMS to be implemented, however, there is currently no approach that can be considered standard across the systems. This can be of primary concern when migration from one CMS product to another is required.

Possible scenarios for CMS migrations range from a takeover by a company or merges of departments within the same company where different CMS have been tailored to individual needs to the case where a company decides to switch to another CMS implementation due to stagnating development progress or inadequate support of the product currently in use. Migration of content requires at least partial integration of CMS and is therefore considered a special form of databases and information systems integration.

This paper reports on the most essential findings of a research project (Nebeling, 2007) designed to investigate migration of content between two of the most popular open-source CMS supporting JSR-170, namely Magnolia (http://www.magnolia.info) and Alfresco (http://www.alfresco.com).

Section 2 of this paper establishes the context of the reported research. Section 3 describes the chosen research design. Sections 4 and 5 present the results as part of the artefact development and evaluation processes of this research. Section 6 presents the reflection involved in the research project. Section 7 draws conclusions based on the experiences gained from this research project.

Nebeling M., Rumantir G. and Smith L. (2008).
MIGRATION BETWEEN CONTENT MANAGEMENT SYSTEMS - Exploiting the JSR-170 Content Repository Standard.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - DISI*, pages 240-245
DOI: 10.5220/0001699802400245

2 RESEARCH CONTEXT

It is often argued in the literature that for content management to be implemented effectively, the focus has to be on the design of the content model (eg. Päivärinta et al., 2005; Grossniklaus et al., 2002). Content models describe the relationships between the most important aspects of content such as container, type, structure, format, and metadata. Relatively little research has been conducted into the design of content models (cf. Tyrväinen et al., 2006).

Grossniklaus et al. (2002) propose separate models of content, structure, and format for content management implementations. While they focus mainly on these components, Päivärinta et al. (2005) require content lifecycles, metadata, and corporate taxonomy to be integrated in the content model. As yet, a unified model that can be implemented to support all of these components has not been presented. In fact, in their analysis of 58 content management projects and implementations, Päivärinta et al. (2005) report "no mentions of explicit, let alone unified, content modelling approaches" (p. 3).

However, with the Content Repository API for Java™ Technology Specification (Nuescheler et al., 2006) considerable work has been done to create a Java standard under the Java Specification Request 170 (JSR-170) that provides the necessary constructs for the implementation of content models. JSR-170 started as a proposal to be accepted as an industry standard for the Java language and related platforms. Members from more than 20 different companies, well-known in the content management field, participated in its development. While this research focused on the recently approved JSR-170, a new version specified as JSR-283 was under public review in September 2007.

JSR-170 provides a hierarchical, tree-based view of the content repository. Here, content is organised in a structure similar to that of a folder and file structure known from the file system, where folders would represent nodes and files would represent the properties of the nodes. For the purpose of classification, each node in the repository must be of a primary node type, and optionally, can add so called mixin node types.

With the system view, JSR-170 also defines a complete mapping of the content repository to XML. System view XML exports and imports can in principal be exploited to facilitate migration of content between CMS based on JSR-170. However, this requires an integrated content model that would resolve the differences between the content models of the source and destination CMS implementations.

3 RESEARCH DESIGN

For this research, a design science approach (Hevner et al., 2004) was chosen because the focus of the research was on the design of an innovative artefact that would facilitate migration between the selected CMS implementations. Similar to Arnott (2006) the research process involved the five distinct phases as outlined in the following subsections.

3.1 Problem Recognition

The first phase of the research project aimed at the identification of differences between the selected CMS in terms of the implemented content models. An experiment was designed in which the same simple website (Figure 1) was created using each product.

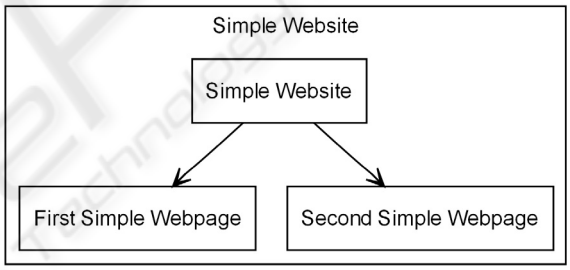

Figure 1: Structure of the simple website.

For the purpose of this investigation, the structure of the website was kept simple and the content plain. Although more complex structures would more realistically represent today's websites, they would largely produce repetitive content structures.

It was decided to concentrate on Magnolia and Alfresco as they are two of the most popular open-source products that support the JSR-170 standard to organise and access the content repositories. Magnolia was the first open-source CMS to be based on JSR-170, and therefore has found widespread acceptance. It uses Jackrabbit as the underlying JSR-170 content repository implementation. Alfresco, on the other hand, implements its own fully compliant JSR-170 repository and supports a range of industry-wide Java standards, which is why it has become a popular choice for many content management projects.

Once the simple website had been created using Magnolia and Alfresco (Section 4.1), the produced content structures were mapped to XML using the

system view export from each system. Also, for each system, the implemented types used for the classification of the proprietary content structures were exported to XML. As the researcher found it difficult to explore and analyse the collected data in its pure XML format, Extensible Stylesheet Transformations (XSLT) were applied to prepare the data for visualisation using the GraphViz tools. This visualisation then allowed the researcher to explore and analyse the data more clearly.

3.2 Suggestion

The investigation conducted in the first step found substantial differences in the content models implemented by Magnolia and Alfresco. To resolve these differences, the researcher proposed an integrated content model based on the individual content models implemented by the selected systems.

Parsons et al. (2000) argue for the integration process that it is necessary to regard "instances and their properties independent of any classification" (p. 239). In a JSR-170 compliant repository, all content "is ultimately accessed through properties" (Nuescheler et al., 2006, p. 40), and nodes storing content units only exist as members of node types. Parsons et al. (2000) propose a two-layered approach to the integration process, which separates the instances or nodes from the classes or node types they belong to. Following this approach, the first layer only represented *the instance model* providing information about nodes stored in the repository and their properties without any classification. The second layer then represented *the class model* providing information about node types only defined in terms of their properties.

The two-layered approach had the advantage that the integration of the content models could be performed at the instance level without first agreeing on a set of node types. As far as the class model is concerned, Parsons et al. (2000) suggest that, once the instance model has been combined, "the class model can include all classes from all sources" (p. 248). However, each CMS implementation then must have at least partial knowledge about the node types defined by the other system. In order for the class model to represent only a global set of classes for both Magnolia and Alfresco, a single generic node type was defined for each set of similar content structures. In making such an abstraction further known as *Generalisation*, the researcher was able to resolve most of the differences between the individual content models implemented by Magnolia and Alfresco. This integrated model was then implemented in Jackrabbit and simple repository transformations were defined to exchange the simple website between the systems (Figure 2).

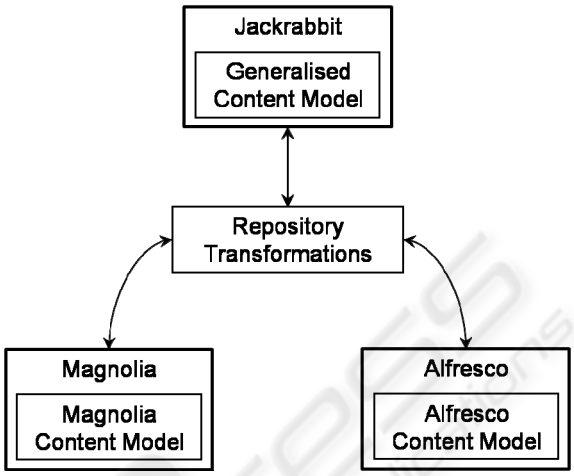

Figure 2: Proposed framework for automated migrations of content between Magnolia, Alfresco, and Jackrabbit using simple transformation methods.

3.3 Artefact Development

The artefact at the core of this research was the *Generalised Content Model* (Section 4.2). This term was chosen to reflect the fact that the proposed model is a generalisation of existing content models whose primary purpose was to provide an integrated view of the individual content models implemented by Magnolia and Alfresco.

Instantiations are used to demonstrate feasibility and are aimed to prove that an artefact can be implemented and executed in a working system (Hevner et al., 2004). The instantiation of the proposed model was in the form of a JSR-170 node type definition implemented in Jackrabbit to facilitate storage and retrieval of content in the Generalised Repository (Section 4.3).

3.4 Artefact Evaluation

The evaluation of the proposed model was conducted in two steps: (1) It was tested if the node structures extracted from the simple website in Magnolia and Alfresco can be transformed to the generalised content structure, and fully restored when transforming back to each source repository. (2) It was tested if the content structures of the simple website created in Magnolia and Alfresco could be interchangeably transformed in both directions between the two systems using Jackrabbit as the Generalised Repository in between.

If Step (1) proved successful then the approach towards the Generalised Content Model was validated for Magnolia and Alfresco (Section 5.1). If Steps (1) and (2) proved successful then the proposed model would also facilitate migration between the selected systems (Section 5.2).

3.5 Reflection

Reflection typically is involved in the final step of design science research (Arnott, 2006; Hevner et al., 2004). For this research project, the chosen research design and the artefact design processes were critically reviewed. As a result, refinements of the proposed model and its instantiation were identified to indicate further research directions (Section 6).

4 ARTEFACT DEVELOPMENT

In this section, the artefact development process is reported in three parts. The first part summarises the findings of the initial simple website experiment. The second part describes the Generalised Content Model that was proposed based on the findings of the initial experiment. The third part reports on important design decisions for the instantiation of the proposed model.

4.1 Experiment Findings

The simple website experiment allowed the researcher to identify substantial differences between the content models implemented by Magnolia and Alfresco. With Magnolia, node types are defined in a rather abstract manner and therefore lack explicit agreement on how content is structured in the repository. Even worse is the fact that with different templates the actual content was sometimes stored with different properties. Also, the Community Edition used for the investigation did not implement a particular metadata standard. Even though only the out-of-the-box node type definitions were used for the investigation, many websites created using Magnolia may be based on exactly these default node type definitions, and the identified problems make migration of content from and to Magnolia difficult.

With Alfresco, on the contrary, there is a rather explicit agreement on how content is structured in the repository, which is primarily due to the extensive use of mixin node types. With Dublin Core, a widely-used metadata standard is partially implemented in Alfresco; however, the system did not make use of the respective elements when the simple website was created and associated with metadata.

4.2 Proposed Model

Based on the findings of the simple website experiment, the Generalised Content Model as illustrated in Figure 3 was proposed.

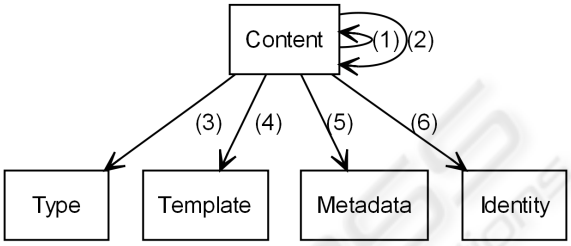

Figure 3: Proposed Generalised Content Model.

The relations between the different components in Figure 3 can be described as follows: (1) With the exception of the root container, each Content unit always has a container. (2) Depending on the chosen level of granularity, each Content unit can contain finer grained units. Together, these relations make for a hierarchy of Content units, typically referred to as the content structure. (3) Each Content unit is always associated with a Type. The Type makes for an explicit agreement on the content structure. (4) Each Content unit is assigned to at least one Template. Templates are used to transform content from one form into another. Each form enriches the Content unit with format; however, the format must have no impact on the content structure. (5) Each Content unit is described by a set of Metadata elements. It is good practice to implement existing metadata models such as the Dublin Core Metadata Standard. (6) Each Content unit can have multiple relations to each Identity that participates in the management of the content lifecycle. The two investigated CMS implementations both assign at least the creator with each Content unit.

The proposed model is not necessarily an alternative to the separate models proposed by Grossniklaus et al. (2002). Rather, it is attempted to integrate the relationships between content, type, structure, and format into a unified view of the content model. Päivärinta et al. (2005) require the content model to support content lifecycles and metadata. Metadata is clearly an essential part of the proposed model but content lifecycles may need further integration. Päivärinta et al. (2005) also emphasise the need for corporate taxonomy; however, the researcher argues that corporate taxonomy can be supported with

metadata provided that a taxonomy of metadata elements can be established.

4.3 Instantiation of Proposed Model

The instantiation of the proposed model is a node type definition for JSR-170 compliant repositories. It was implemented and validated with Jackrabbit.

Mixin node types were first defined to establish the difference between nodes storing content and metadata in the repository as well as to allow for a hierarchical structure of content units and a taxonomy of metadata elements. The use of mixin node types yields the advantage that nodes storing actual content can inherit from any particular primary node type but will still be marked as either content or metadata, simply by adding one of the proposed mixin node types. As a result, the generalised node types can be integrated with existing node type definitions more easily.

Simple primary node types for the storage of the actual content were then proposed as a generalisation of the individual node types implemented by Magnolia and Alfresco. As far as metadata is concerned, the Dublin Core simple metadata element set was implemented since it was sufficient to describe the content of the simple website; however, no particular standard is inherent to the proposed model.

5 ARTEFACT EVALUATION

In this section, the results of the artefact evaluation processes are presented in two parts. The first part concentrates on the simple repository transformations that were defined to validate the proposed model for Magnolia and Alfresco. The second part reports on the exchanges of the simple website between the two systems in both directions.

5.1 Repository Transformations

Four distinct transformation processes between Magnolia, Alfresco, and the Generalised Repository were differentiated: (1) from Magnolia to the Generalised Repository, (2) from the Generalised Repository to Magnolia, (3) from the Generalised Repository to Alfresco, and (4) from Alfresco to the Generalised Repository. In all four cases the Generalised Repository was Jackrabbit (Figure 4). Simple transformation processes based on JSR-170 were defined for the exchange of the simple website between each repository.

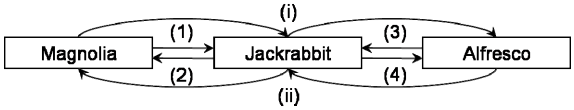

Figure 4: The repository transformation processes between Magnolia, Alfresco, and Jackrabbit.

Processes (2) and (4) were, in fact, inverse transformations of Processes (1) and (3), respectively; however, exceptions had to be made because with Magnolia's templates the content was sometimes stored with different properties and, for Alfresco, the implemented mixin node types had to be fully restored as part of Process (4).

The distinct repository transformations were implemented in separate Java classes and, while the crucial steps involved in the repository transformation processes were logged, were exercised as follows: The simple website was transformed through Process (1) and fully restored through Process (2), i.e. without substantial loss of information. As a result, the proposed model was validated for Magnolia. The same was shown for Processes (3) and (4), respectively, so that the proposed model was also validated for Alfresco.

5.2 Migration of the Simple Website

Final experiments exercised the following compositions of repository transformations: (i) from Magnolia to the Generalised Repository and from the Generalised Repository to Alfresco, and (ii) from Alfresco to the Generalised Repository and from the Generalised Repository to Magnolia. Again, the Generalised Repository was Jackrabbit in both cases (Figure 4). Hence, running Composition (i) demonstrated migration from Magnolia to Alfresco; running Composition (ii) demonstrated migration from Alfresco to Magnolia, using Jackrabbit as the intermediator in both cases.

The correct display of the simple website after executing Compositions (i) and (ii) together with the visualisation of the exported node structures and the repository transformation protocols provided strong evidence that the simple website was effectively exchanged between Magnolia and Alfresco in both directions. As a result, the initial hypothesis, that the Generalised Content Model can resolve individual differences between the selected systems and therefore facilitate migration of content between them, was accepted.

6 REFLECTION

Design science is research into building artefacts that address "important unsolved problems in unique or innovative ways" (Hevner et al., 2004, p. 81). The artefact at the core of this research is a model-based approach towards the migration of content between two CMS based on JSR-170. It is unique and innovative because, to the knowledge of the researcher, no similar approach has been presented.

In the initial phase of the artefact development, the two content models implemented by Magnolia and Alfresco were studied. An important step was to extract and scrutinize the different node type definitions implemented by the two selected systems. As there was no documentation of the node type definitions available, some of the findings of the simple website experiment had to be tentatively explained and may need further clarification. Therefore, the researcher wishes to emphasise the need for a documentation of content models, where CMS developers may explain and justify their design decisions.

The design of the proposed model was based primarily on the most essential commonalities and differences between Magnolia and Alfresco as identified in the simple website experiment. Hence, the researcher notes critically that the proposed model is an approach towards *a* Generalised Content Model rather than *the* Generalised Content Model, and still is only based on Magnolia and Alfresco.

7 CONCLUSIONS

Content management systems (CMS) have evolved in diverse ways due to the lack of sufficient standards. Only recently, with the JSR-170 content repository standard, an ongoing effort to standardise CMS can be observed. The reported research project was designed to investigate migration of content between two of the most popular open-source CMS supporting JSR-170, namely Magnolia and Alfresco.

An initial experiment where the same simple website was created using each of the selected systems found incompatible content structures due to differences in the content models implemented by Magnolia and Alfresco. To resolve these differences, the Generalised Content Model was proposed as an integration of both content models, and implemented in Jackrabbit, the JSR-170 reference implementation. Simple transformation methods were then defined to transform the content structures between Magnolia, Alfresco, and Jackrabbit. Final experiments provided evidence that the simple website could effectively be exchanged between the selected CMS in both directions using Jackrabbit as an intermediator.

The research project showed that many individual differences between CMS can be traced back to the lack of a common content model. The relations between content, type, structure, format, and metadata need to be clearly defined to allow for the design of a common content model. With the chosen approach, the researcher was able to resolve individual differences between the content models implemented by Magnolia and Alfresco, and as a result enabled migration of a simple website between the two systems in both directions. This research may therefore be regarded as a starting point to define methods for automation of migrations between CMS based on JSR-170 in the absence of a common content model. However, the researcher believes that this research is also a blue print for a method to design and evaluate a content model that may become standard across CMS implementations.

REFERENCES

Arnott, D. (2006) Cognitive biases and decision support systems development: a design science approach. Information Systems Journal, 16, 55-78.

Grossniklaus, M. & Norrie, M. C. (2002) Information concepts for content management. Proceedings of the Third International Conference on Web Information Systems Engineering (Workshops).

Hevner, A., March, S. T., Park, J. & Ram, S. (2004) Design Science in Information Systems Research. MIS Quarterly, 28, 75-105.

Nebeling, M., 2007. *Towards a Generalised Content Model: Facilitating Migration between Content Management Systems based on the Content Repository API for Java Technology Specification (JSR-170)*, Monash University. Honours degree of the Bachelor of Information Technology and Systems.

Nuescheler, D. & Piegaze, P. (2006). "Content Repository API for Java™ Technology Specification." Retrieved May 15, 2007, Available from http://www.jcp.org/en/jsr/detail?id=170.

Päivärinta, T. & Munkvold, B. E. (2005) Enterprise Content Management: An Integrated Perspective on Information Management. Proceedings of the 38th Hawaii International Conference on System Sciences.

Parsons, J. & Wand, Y. (2000) Emancipating Instances from the Tyranny of Classes in Information Modeling. ACM Transactions on Database Systems, 25, 228-268.

Tyrväinen, P., Päivärinta, T., Salminen, A. & Iivari, J. (2006) Characterizing the evolving research on enterprise content management. European Journal of Information Systems, 15, 627-634(8).