

One Click Archiving - A Simple System for Bridging the Content and Records Management System Gap

Jan ASKHOJ, Mitsuharu NAGAMORI, Shigeo SUGIMOTO

School of Library and Information Science, University of Tsukuba
Graduate School of Library, Information and Media Studies, University of Tsukuba
E-mail: s0621343@ipe.tsukuba.ac.jp, {nagamori, sugimoto}@slis.tsukuba.ac.jp

Abstract

Content Management Systems (CMS) are widely used for organizations to publish information, to keep transactions and records, and so on. However, in general, CMS in use today do not offer the required level of functionality for an organization that needs to ensure safe, legally compliant records management. It therefore becomes necessary to transfer the records to be retained to a Records Management System (RMS). CMS and RMS are seldom interoperable out of the box, making archiving of retained records difficult. Up to now, the solution to these problems has been to add records to the archive by hand, or to create custom programs for records transfer. Neither of the above solutions is optimal. In this paper, I propose a lightweight approach to the problem of integrating CMS and RMS software. My approach is based on a three layered model for the organization of an corporate records management system. The model allows the connection of one or more CMS to a RMS by making it possible to automatically transfer and ingest retained records for archival.

Using the above model, I have developed ATLAS (Automated Transfer Lightweight Archive System). ATLAS is designed to connect multiple CMS with different metadata schemes to a single RMS, enabling automatic archiving of records submitted by users. Each CMS is registered in ATLAS, along with a metadata crosswalk that translates CMS metadata into a metadata format that can be imported into the RMS. ATLAS also supports registration of additional CMS by allowing administrators to upload metadata crosswalks in XML/OWL. ATLAS uses RSS 2.0 as a protocol for transferring records and metadata. Because it uses open protocols and technologies, such as RSS and XML, ATLAS is designed to work with existing organizational CMS and RMS. With this paper I hope to have shown that the cost of Records Submission in an organizational setting can be significantly reduced by using the three layered model, exemplified by a system like ATLAS.

Keywords

Metadata Schema, Metadata Crosswalks, Records Management, Content Management, Data Transfer

ワンクリックアーカイビング - コンテンツ管理と記録管理のシステムを橋渡す

ヤン アシコイ、永森 光晴、杉本 重雄

筑波大学図書館情報専門学群、筑波大学図書館情報メディア研究科

E-mail: s0621343@ipe.tsukuba.ac.jp, {nagamori, sugimoto}@slis.tsukuba.ac.jp

概要

企業や組織内では情報の作製や公表のためにコンテンツ管理システム (CMS) がよく使用されている。現在では、米国のサーベンス・オクスリー法といった企業経営者の責任と義務・罰則を定めた法律に従って、組織や企業が持つ情報を記録することが求められている。しかし、CMS と RMS の連携は考慮されておらず、現在用いられている一般的な CMS の機能では、法律に従った記録管理のために、CMS 上の情報を記録管理システム (RMS) に安全に転送することは難しい。一般的には手動で情報を CMS から RMS に移動させる、または独自の転送プログラムを作るという手間のかかる方法で解決されている。そこで本研究では、CMS と RMS を結び付けるために三層モデルを提案する。このモデルでは、組織内のアーカイブの構造を三層に分けて表現し、異なる CMS が持つ記録すべき情報を RSS や Atom を利用して RMS に記録する。本稿では、三層モデルと、そのモデルに基づいて開発した ATLAS (Automated Transfer Lightweight Archive System) について述べる。

キーワード

メタデータスキーマ、メタデータ Crosswalk、レコードマネジメント、コンテンツマネジメント、データ転送

1. Introduction

In companies and organizations, there is a need to archive business critical content, regardless of form or media, in a secure consistent manner. There are many reasons for this, such as safeguarding vital information, improving efficiency and productivity by helping search and retrieval. Recently, following the Sarbanes Oxley Act and other legal measures in the US, having a proper records management program in use is vital for ensuring regulatory compliance. To ensure this, most larger companies/organizations need records management systems with specialized functionality which can support the records management process [1]. Marcel Robles and Mark Langemos defines Records management as:

“The professional management of information in the physical form of records from the time records are received or created through their processing, distribution, and placement in a storage and retrieval system until either eventual elimination or identification for permanent archival retention.” [2]

While some systems exist that can manage content and records throughout the entire content lifecycle, this is far from the norm (I define a record as a piece of content that has reached its final form, and that is deemed archive-worthy according to the organizations retention schedule). Implementing a system with functionality to handle different types of content throughout all stages of its lifecycle is a costly endeavor. Organizations that want to avoid this and still need specialized records management functionality therefore need to transfer records from where they are created and used to a RMS. Because of the large amount of records in organizations today, as well as the complexity of the many systems involved, the process of getting records in to the RMS can be a problem [3].

To make matters worse, many organizations today use a number of different tools to help with the content creation and management process, depending on the task. Examples of this are Document Management Systems, Web content management Systems, Digital Asset Management Systems etc. All of these systems are examples of Content Management Systems (CMS).

While there is no canonical definition of what a CMS is, I use the following definition for content management:

“Content management encompasses a set of processes and technologies, enabling the creation and packaging of content (documents, complex media, applets, components, etc.) as part of a dynamic and integrated Web-centric environment.” [4]

In this paper, a CMS is a system that supports the creation and management of a wide range of content types as described above. If this definition is compared to the definition of Records Management, it becomes clear that the focus for these two concept is on different stages of the content lifecycle, as illustrated in figure 1.

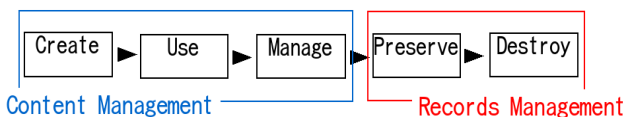


Figure 1: CM and RM in the content lifecycle

Common for CMS is that they store content that needs to be retained in a secure manner, according to archiving policy. This means a situation where organizations are left with important records in a number of different systems, that are not designed with archiving in mind, and that have insufficient

functionality to guarantee the safe, long term preservation of records.

Therefore, a simple model that integrates both content creation from multiple sources, as well as archiving functionality is highly desirable. The benefits of such a system would be:

- Organizations would be able to use whatever CMS systems needed, without worrying about records management. This would be less costly than buying or building specialized systems containing both CMS and RMS functionality for all the content types in use.
- Users would not have to spend their time manually moving their records from CMS to RMS.
- Organizations would not have to buy or develop custom solutions for moving content from CMS to RMS.

In other words, there is a real need for a lightweight model for the transfer content from CMS to RMS. Lightweight in this context means easy to implement, manage and change.

To achieve this goal of simplicity, it is important to keep the number of functional entities in the model as low as possible.

I seek to achieve this goal of integrating CMS and RMS by proposing a new model for integration. Having proposed this model, I plan to use it as the basis for building a system that allows the automated transfer of content from one or more CMS to a RMS.

2. Background

2.1 Problems with existing methods of records transfer

There already exist a number of models for transferring electronic records to a repository or archive. Models such as the OAIS [5] are generally applicable, very detailed, and offer a high degree of functionality. However, these kinds of system are aimed at archival institutions or large digital archive systems, where the scope is much broader than in a typical private organization. In smaller organizations, using a lightweight approach to the problem would be preferable to adopting such a comprehensive model. Therefore the solution has often been to transfer records deemed archive-worthy by hand, or to use custom systems for records transfer that are designed to match the software and hardware profile of the organization.

Both of these methods are problematic. Transferring records from one system to another by hand is both time consuming and error-prone. A typical workflow might look like this:

- A user or a group of users is working on a content item in a CMS. Once the content is finished, the user saves it in the CMS and marks it “final version”. Since the content has to be archived according to the organizational retention schedule, the user saves a copy of the content to a local folder on his PC or on the network. The user then logs in to the organizations RMS and navigates to the “upload content” screen. Here the user fills in the required metadata fields by hand and uploads the content to the RMS.

And of course, the larger the amount of records and metadata, the more costly the process becomes for the organization.

Building specialized programs for transferring records, such as ad-hoc export-import scripts also has drawbacks: Since the programs have to be custom-built to fit the systems already in use in the organization, in-depth knowledge about these system is required. Furthermore, custom built programs or specialized adaptations of existing programs can be expensive to develop. And finally, in case of a system migration or changes to the application programming interfaces (API) used

on either the content creation side or records management side, the programs might have to be rewritten. This may also be necessary if there are changes to the organizational metadata schema, or if a new CMS is added to the program.

Getting digital content from multiple sources into a common repository is in some ways similar to existing Web Archiving projects, where the object is the ingest of content in order to ensure long-time preservation (for example The Internet Archive) [7]. One of the benefits of this approach is that it eliminates the cost of manually submitting content and filling in metadata, by reading and importing selected content automatically. On the other hand, a problem with crawler-based web archiving is that dynamically generated content (such as personalized pages) and unless metadata is immediately available in the code read by the crawler, it may not be captured.

In corporate records management, it is also desirable to eliminate the submission process. After all, the time spent submitting content manually costs money. But unlike web archives, it is absolutely essential that all published information to be archived is captured completely and in a format true to the original. Luckily, in a private company or organization, content to be archived is captured from known sources. In other words, the CMS used as source for the ingest, while they may be different, are all within the control of the organization in question. This makes it possible to make a model that where all the entities (CMS, RMS and the connecting data transfer) can be adapted to fit the organizations needs. However, from the standpoint of an organization, it is desirable to make as few adaptations as possible, since all changes to existing systems means extra cost.

Another problem of conventional records transfer methods is metadata. Because of the differences between Content and Records Management, the metadata schemas used are likely to differ as well. This problem only gets worse when transferring content from multiple sources, in which case you have to deal with several different metadata schemas. The need to design a model of interaction between the CMS and RMS comes from the fact that there are differences between the two types of system, and as a consequence, they may not be able to exchange data out of the box. In a CMS, managed content can be made up by any combination of elements from a number of sources, formatted according to display preferences.

These facts cause a number of problems when wanting to add the content to a generic RMS.

- First of all, not all content in a CMS should be archived, so a records declaration process needs to take place.
- The content may not be in a format that can be added directly to the RMS. In order to ensure long time preservation, some organizations may wish to convert content into different file formats in order to ensure long-term usability, e.g. from Microsoft Word to PDF.
- Because of the inherent differences between content management and records management, metadata selection and conversion from one schema to another will most likely be necessary. Where schema conversion is concerned, three scenarios can be imagined. (1) The necessary metadata in the CMS fits the RMS metadata. (2) The necessary metadata exists in the CMS, but doesn't fit the RMS metadata schema. (3) The metadata needed by the RMS doesn't exist in the CMS. Of these three scenarios, 2 and 3 will need to be addressed for successful metadata transfer to take place. If more than one CMS is used in the model, the conversion between schemas becomes $N \rightarrow 1$, as for example a CMS for

Digital Rights Management may have a totally different set of metadata compared to a CMS for Web Publishing. (It is possible to imagine an organization with several different RMS ($N \rightarrow 1$), but this scenario is very rare, and is considered outside the scope of this paper.

- While it is true that most current CMS and Records Management Systems support web protocols such as HTTP and FTP, they are not designed to exchange data in a way that allows easy transfer and ingest of records.

2.2 Metadata crosswalks and data loss

Ideally, a transport package would be able to transfer CMS metadata to the RMS in its original format. But as explained in chapter , the metadata schema in use in an organizational CMS and RMS are likely to be different. In this case, some kind of metadata crosswalk becomes necessary to change the CMS metadata so that it matches the RMS metadata schema. There are a number of existing crosswalks available, such as FGDC to USMARC and ADL to FGDC [8]. In order to handle differences in metadata schemes, it may be necessary to add metadata crosswalking to the a system built on the three layered model.

However, when crosswalking/translating from one schema to another, the result is almost inevitably loss of data. In "Crosswalks The Path to Universal Access?", Mary S. Woodley lists six common misalignments between schemas that serve as a cause of data loss: [9]

1. *A concept in the original database does not have a perfect equivalent in the target database*
2. *Data that exists in one field in the original schema may exist in separate fields in the target database.*
3. *Data in separate fields in the original schema may be in a single field in the target schema.*
4. *Information in one schema may reside in a field that is indexed, whereas it is only free-text descriptive information in the other schema.*
5. *There is no field in the target schema with an equivalent meaning, and unrelated information may be forced into the same "bucket."*
6. *In only a few cases does the mapping work equally well in both directions. (See number 2 above.)*

To increase the usability of a crosswalk, it is desirable to keep the data loss to a minimum. In the case of ATLAS, some steps have been added to avoid data loss, but more could no doubt be implemented given additional time. In order to give an idea of the effectiveness of the metadata crosswalk functionality in ATLAS, I have applied Woodleys six common schema misalignments.

1. *No perfect equivalent in the target database.*
This problem also exists in ATLAS, but is not related to its functionality. It is exclusively a mapping problem.
2. *Data in one field may exist in separate fields in the target.*
Functionality to deal with this problem has been built into ATLAS. For example, the CMS element type can be split into Type and the qualifier Record Type
3. *Data in separate fields may be in a single field in the target schema.*
Functionality to deal with this problem does not yet exist in ATLAS.
4. *Information in one schema may be in an indexed field, but in a free-text field in the target schema.*
This problem depends more on the indexing policies of the used CMS and RMS than on functionality in ATLAS. That said, it can still be a problem.

5. There is no field in the target schema with an equivalent meaning, and unrelated information may be forced into the same "bucket."

This is also a mapping problem.

6. In only a few cases does the mapping work equally well in both directions.

This problem lies outside the scope of ATLAS, as it is only designed for one directional mapping (CMS → RMS)

As can be seen from the above points, a lot of the crosswalk problems are due to field mapping, but some are more a question of functionality, such as field splits and merges. Either way, in order to ensure that no metadata is lost in the translation, ATLAS sends a copy of the original CMS metadata to the RMS, where this can be saved as a separate item in the record. This should help subsequent data retrieval by at least making it possible to search the old metadata as full-text.

Finally, to add to the schema problems outlined by Woodley, there is the problem of element values. Some RMS may have specific requirements for the format of the data values received from ATLAS. Whereas there is no support for such functionality in ATLAS at the moment, various proprietary and open-source data format converters exist, for example WCSTools Getdate. [10]

2.3 Requirements for a records transfer model

Analyzing the previously presented problems with existing methods of records transfer, a number of requirements for a new transfer model can be specified.

- The model must support the automated transfer of records. As mentioned previously, a manual records submission process requires the user to perform a long series of steps in order to transfer a record to the RMS. Ideally, the records transfer process would be totally invisible to users and archivists. As soon as content is finalized, it should be transferred to the RMS for proper storage.
- The model must be lightweight. Records management is a business process, not the sole purpose of the business (as compared to web archives and traditional historical archives), this means that the model must only contain the required functionality for successful records transfer, and make as few demands on the organization as possible.
- The model must be flexible enough to work with a wide range of systems. There are a large number of CMS and RMS on the market, with varying degrees of functionality. The model must be designed to work with as many of these as possible. Furthermore, some organizations may use more than one CMS. It must be possible to transfer records from multiple CMS.
- Finally, the model must support the transfer of metadata. Since CMS also use metadata to support the content management process, it must be possible to reuse this metadata where applicable.

3. Model for Connecting CMS and RMS

3.1 Introduction to the three layered model

Based on the requirements in chapter 2, I have developed a new model for integrating CMS and RMS functionality to create a complete archiving solution. My model differs from existing models in that it works with almost any type of off-the-shelf system with only a few modifications, while still being lightweight. It also solves the problems of content transfer and metadata conversion, and offers corporate

archives an automated approach to creating and managing content in native systems, while allowing them to remain compliant with records management regulations.

3.2 The Three Layered Model explained

Figure 1 gives an overview of the three layered model. The left side represents the CMS and the right the RMS.

The first layer is the content layer. Content is produced or imported into the CMS, where it will most likely undergo a number of revisions. When the revised content is eventually published in one form or another, it will be transferred to the RMS.

The second layer represents the metadata about the content. Metadata will almost certainly be different in the CMS and RMS. Generally speaking, in a basic CMS, the focus is on managing and producing content. The CMS support this process by offering functionality such as versioning, keywords etc. All of these functions can be described as different types of metadata, such as descriptive metadata, administrative metadata and structural metadata. This metadata is all related to the content which it describes and is particular to the type of CMS in question. In an RMS, the focus is on long time storage of static records, and the metadata in use in such a system is tailored to these needs, so metadata is used that supports functionality such as retention and disposal management, record series management etc. [11]

The third layer is the meta-metadata Layer which contains the data necessary for communication between the CMS and the RMS. This layer has several functions.

- The first function is to ensure that all relevant data is made available for transfer.
- The content and metadata is transferred using suitable formats which can be read by the RMS.
- Finally, the third layer ensures safe transfer of data from one system to another

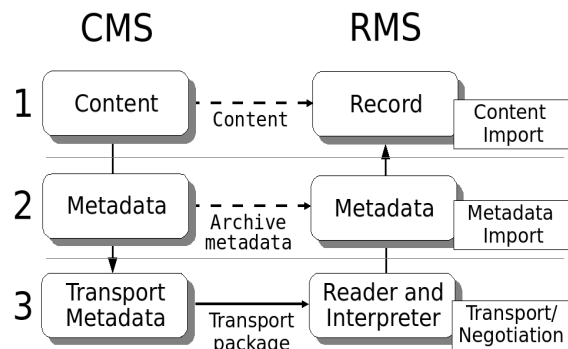


Figure 2: The three layered model

The arrows in the model represent the flow of information. This will be described in further detail in chapter 4, but in short: Content and metadata from CMS layer 1 and 2 is sent by CMS layer 3 to RMS layer 3. After import into the RMS, it resides in RMS layers 1 and 2. In other words, the CMS layer 1 content becomes RMS layer 1 records and the CMS layer 2 metadata becomes RMS layer 2 metadata. This connection is shown by the horizontal dotted lines in layer 1 and 2.

I have designed the three layered model to be as simple as possible, while still ensuring that all archive-worthy content and metadata is transferred safely to the RMS. By simple I mean:

- It has only a few entities.

- It works with off-the-shelf/existing tools.
- It uses simple protocols.
- It is suited to a specific situation, i.e. building an archiving system by integrating CMS and RMS.

These factors form the basis of my approach, and will help facilitate the implementation of a system designed using the three layered model in an organizational setting.

4. Information flow in the three layered model

Unlike traditional archives and public electronic repositories, the main purpose of corporate records management is to support business processes and policy, enabling organizations to handle their own records effectively and in accordance with law. This lends some unique properties. Among these are: Shorter retention periods, the use of original document formats rather than conversion, automated workflows, and finally a high degree of integration between systems, procedures and tools [12]. This integration is made possible by the aforementioned handling of the entire records lifecycle within a single organization.

Any model to be used in an corporate setting must take these properties into account, as they have an impact on the records management process. To illustrate how the model would work in practice, I have prepared an example of information flow.

The data flow in the model (figure 1) is from left to right. In theory, the information in one layer corresponds to the information in the same layer on the opposite side of the model (e.g. CMS content to RMS content). However in reality, the data flows from CMS level 1 and 2 down to level 3, is transported to RMS level 3 and passed up to level 2 and 1. During this process, there may be changes to the data, such as conversion or selection, but it is still the same data entity (e.g. content is still content).

4.1 Content Management System

The left part of the model represents the CMS, split in three layers. The model can operate with input from multiple sources, so the left side can be thought of as several CMSs, all represented using the three layer model.

CMS Layer 1 - Content

In the CMS, a new post is created as either as new content or as content reusing one or more existing elements. The document is revised and updated, using content management functionality in the system. At some point, a version of the content considered final will be published or used for some business purpose. If the content type is marked as a retained record in the organizational retention schedule, the content needs to be archived.

CMS Layer 2 – Metadata

At the time of creation, some metadata will be created automatically (publishing status, creator name, time of creation etc), and some metadata may be added manually (keywords, subject etc). CMS functions such as versioning, check in/out, also add information to the content, and as such generate metadata.

All of this metadata is managed according to the metadata schema used in the CMS. Some of this metadata may be relevant when archiving the content, while some may not.

Apart from the metadata used by the CMS to manage content, most content types themselves also contain metadata of some sort. For example, Microsoft Word documents have built in property fields for keywords and comments. This metadata will be transferred along with the originating file, and not be converted or otherwise managed by an ATLAS system. The

reason for this is that because of the relative short retention schedules in use in corporate RMS, documents or spreadsheets are mostly stored in their original format. This extends to any document-native metadata.

CMS Layer 3 - Transport

Level three is responsible for selecting the content and metadata, and transporting it to the RMS for ingestion. In the case of content, selection happens according to parameters from the company retention schedule. For example, all content tagged “contract” must be archived, and will be passed to layer 3. In the case of metadata, all available metadata from the originating CMS post is included in the transport package, and will be read and converted by the RMS Level 3 Reader and interpreter.

A transport package contains three elements: Content, Metadata, Transport Metadata. The elements of the package are made up of:

- Content: Content is actually an electronic copy of the post in the CMS. In the system, all the data that makes up the competent package will be stored in one or more databases. Content can be stored as plain text, HTML and mime-type file attachments such as Word or PDF.
- Metadata: Metadata is similarly stored as fields in a database, and the format and values it takes will depend on the metadata schema used for the CMS.
- Transport package: On the CMS side, The transport package collects and packages the data to be sent to the RMS, namely the Content, Metadata and transport metadata. Transport metadata is generated by layer 3 to ensure safe and successful data transfer, and to help with metadata conversion. When a package is made available, it includes a metadata-schema ID for the metadata it contains.

4.2 Records Management System

The right part of the model represents the RMS part of the system, also split in three layers.

RMS Layer 3 – Transport

On the RMS side as well, the transport layer has several functions:

- It regularly receives transport packages from the CMS and saves them to a temporary location.
- Using the Metadata Authority URI, the Reader and interpreter can acquire a conversion table for the metadata schema used in the received Transport Package. Metadata that is not needed by the RMS is ignored. Metadata fields that are needed by the RMS but not provided by the CMS are left empty.
- The transport layer is also responsible for logging the receipt of a transport package, and sending an acknowledgment back to the CMS to show the package has been received.

Layer three has an additional important function: To decide if content should be archived or not. Most corporate archives use Retention Schedules (also known as File Plans) to determine what content should be archived. These schedules are divided by record type, and contain detailed instructions for archiving. In the model, archiving is also decided by record type. In other words, all finalized content of a certain type must be archived. When a CMS metadata schema is registered in the Metadata Authority, whatever field in it that defines the Content Type will also be registered, and mapped to the corresponding Archive Metadata field for Record Type. If the value in the Record Type field matches a Record Type to be archived according to the Retention Schedule, the Content will be

archived. If the Content type is not mentioned in the Retention Schedule, the content in question, along with its metadata will not be added to the RMS.

RMS Layer 2 - Metadata

Before the converted metadata from the transport package is added to the RMS together with the corresponding content, it needs to be reviewed by an archivist. This is to ensure that no incorrect metadata is added to the RMS, but also to manually fill in any fields with no metadata (where there is no corresponding metadata in the CMS). Depending on what metadata schema is in use in the RMS, the transport package itself may also hold information that can be used as metadata in the RMS, an example of this is the date-time the transfer package was added to the RMS.

The metadata will be imported using the import tools built into the Archiving software.

RMS Layer 1 - Content

The content and metadata from the data package will be ingested after the archivist review. In the model, content will be added in the same format as it was in when it existed in the CMS. The content will be ingested using the import tools built into the archiving software.

5. Related models

Setting aside the number of ad-hoc solutions in existence, there are already a number of models that deal with records transfer and ingest. A well known example of such a system is the OAIS (Open Archival Information System). OAIS has achieved its popularity based on a number of factors. It is an ISO standard, it is completely open, it is well documented, and it has been in effect since 2002 with parts of it being implemented in the real world [13].

The OAIS model may be applicable to any archive, and contains recommendations for a number of archival information preservation functions including ingest, archival storage, data management, access, and dissemination.

There are a number of differences between the OAIS and the model I am proposing. First of all, OAIS and ATLAS differ in their approach. The object of OAIS is to describe an entire archive system, whereas ATLAS deals with integrating generic CMS and RMS, to make an archiving system without defining these entities in detail.

One of key points of the OAIS model is that it is focused on long time preservation. This is handled by the Preservation Planning entity, which functions include evaluating the contents of the archive and periodically recommending updates to the archival information and guaranteeing data accessibility even if the original computing environment becomes obsolete. Another characteristic of OAIS is the large number of functional entities and subsequent high number of functions provided. An example of this is the Ingest entity, which has five sub-functions, each responsible for carrying out a number of actions. This makes OAIS a very complex model its entirety. In an corporate setting, where records management is just one of many business functions that need to be carried out, only a subset of the OAIS would be needed to build a workable archiving system. Also, the high level of functionality in the different OAIS Entities and the interaction between these, make the OAIS unsuited for implementation with the simple systems and procedures in use in organizations not specializing in archiving. [14]

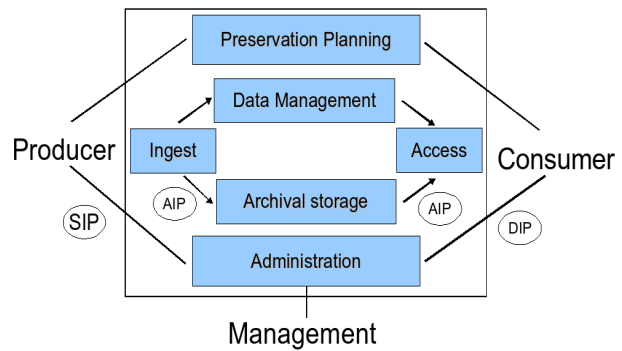


Figure 2: The OAIS model and its main entities

In such a setting, where the focus is on integrating off-the-shelf systems, the ATLAS model has an advantage in the fact that it makes little demand on the CMS and RMS and their functionality. It can also be kept comparatively simple by not using Information Packages that explicitly define Preservation Description Information. Finally, unlike OAIS, records do not necessarily need to be independently understandable (understandable without the assistance of the content producers). It is not uncommon for records in a RMS to be of no use to anyone but the producers, but they may still need to be archived because of legal, contractual or other obligations.

Another model that shares similarities to the three layered model is the The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). This protocol is also built on existing tools, such as HTTP and XML, and harvests metadata of any format specified by a community, based on unqualified Dublin Core [15][16].

OAI-PMH is useful as a protocol, and could with some modifications be used in a system built on the ATLAS model, but as it is designed for metadata harvesting, it cannot transfer content. Furthermore, as a harvesting protocol, it does not in itself contain functionality for reporting back to the data provider whether the data transfer has completed successfully. Finally, as OAI-PMH stands today, specialized metadata schemas is supported, but the protocol requires that servers do use unqualified Dublin Core metadata in XML as a basis. ATLAS makes no such requirement.

6. Building the ATLAS system

Based on the three layered model, I have developed ATLAS. Before moving on to a detailed description of the implementation, this chapter will present an overview of the main elements and how the system works.

Figure 3 shows an overview of the ATLAS system. The three main elements of the ATLAS system are:

1. One or more CMS with support for RSS and the RSS extensions described under "Technical Overview". The CMS could in principle be of any type, as long as the content can be embedded in an RSS feed. Since the object of ATLAS is to store business critical records, all embedded content must be transferred to the RMS to guarantee the completeness of the record. This means that at the time of publishing, the CMS must simultaneously publish (make available for the export module) all the content that is needed to make a complete record (intellectual unit).
2. In the ATLAS system built for this thesis, the CMS is based on Drupal, and has been expanded in order to use extra metadata fields. The CMS supports RSS 2.0, and any

content published will be made available for harvesting in a RSS feed, along with the CMS metadata. The CMS also supports attachments, which are represented as URLs in the feed.

3. An RMS with import functionality supporting metadata in XML. As mentioned previously, metadata coming from the Metadata Authority is formatted as XML, and the RMS needs to be able to read this during the import process.
4. I have used a Dspace digital repository to represent the RMS for this thesis. I have made no changes to the default Dspace installation, other than creating a new collection for storing records and adding metadata elements from the UK-GOV Metadata Standard.
5. The ATLAS Add-ons, consisting of a customized Feed Reader, Metadata Authority, metadata crosswalks and a registration/metadata crosswalk import function.
6. These components are tied together via a Ruby on Rails web-application that stores CMS and Metadata crosswalks in a web-accessible SQL database.

The ATLAS Add-ons are used to connect the CMS and RMS by reading the CMS feed, using metadata crosswalks to translate the metadata and then passing the converted metadata to the RSS for import. In detail, what happens is the following:

1. The Feed Reader reads the feeds from the CMS. In case of a CMS with a badly formatted feed (such as with the customized Drupal system), ATLAS will strip formatting and other unneeded information from the feed.
2. The Metadata Authority splits the feed into individual content items. It then determines whether the content item has already been converted on a previous occasion. If this is the case, it proceeds to the next item.
3. The Metadata Authority extracts metadata from each content, and using the registered crosswalks, the extracted CMS metadata elements fields are converted into the corresponding RMS element fields. For example, the CMS value "date" is translated into "Date.Created".
4. The translated RMS metadata fields are formatted for import into the RMS and stored in the temporary Download Folder.
5. If necessary, an approval process can take place.
6. The records and metadata is imported automatically into the RMS at fixed time intervals.

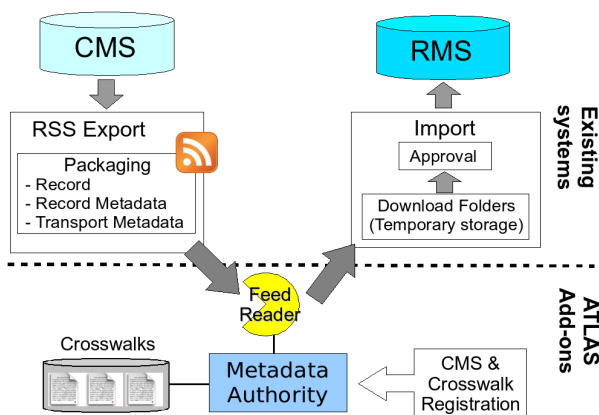


Figure 3. The ATLAS workflow

Finally, ATLAS also supports adding new CMS and crosswalks. This is done in the CMS & Crosswalk registration module, described in detail in chapter 6.2.

6.1 Transport package protocol

In order to transfer records and their metadata from CMS to RMS, a protocol for data exchange is needed. In order to function with ATLAS, there are a number of requirements. First of all, the protocol must allow the safe transfer of both records and metadata. It must be expandable in such a way as to be made to work with multiple metadata schemas. It must be generic enough to be expected to work with a large number of different CMS and RMS. And finally, it must be easy to implement.

The protocol used to transfer content from the CMS to the RMS is RSS 2.0 (Really Simple Syndication) and ATOM. Both protocols are XML based lightweight formats used to publish frequently updated digital content on the internet. It works by an information publisher providing a content "feed" (a machine readable list of published content) on a web site. This feed contains both an amount of content (sometimes only a preview, such as the first 10 lines of an article) and basic metadata. The feed can be downloaded at set intervals by a feed reader/aggregator and read online or off-line without the user having to visit the originating site. There are a number of different, more or less compatible versions of RSS in ATOM in use on the net today. I have chosen RSS 2.0 and ATOM because they are both widely supported and can be extended via namespaces [17].

To explain how protocols work in ATLAS, I will be using RSS 2.0 as an example in the following chapters.

In order to use RSS for sending transport packages from CMS to RMS, it is necessary to add a few enhancements. The first of these enhancements is the support for adding content. In RSS 2.0 feeds, content is written in the <description> field in RSS Item, which can be filled with plain text giving an item synopsis. However, in a CMS, the content to be archived is not limited to text. It may be html, a PDF document, or some other kind of rich content. In such cases, the content can be linked to via an <enclosure> field, also included in RSS Item (this is how RSS is used in Podcasts). The need to enhance RSS content handling arises in those cases, where a single item in a CMS contains several independent binary files. In such cases it may be necessary to add additional <enclosure> fields.

The second necessary enhancement is metadata. As RSS 2.0 only contains a default set of metadata elements, it is necessary to add extra fields to contain the metadata elements used in the CMS. In ATLAS, these extra elements are included in RSS Item.

Finally, it is very important that the <link> field in RSS Channel is used correctly, as this identifies the originating CMS during metadata conversion [18].

The use of RSS in ATLAS

RSS as it is implemented in ATLAS, is not very different from default RSS 2.0. The only place that needs to be extended is the Item element, where custom metadata elements will be added. This results in a structure like this:

- RSS Header - No change compared to default RSS 2.0
- Channel - No change compared to default RSS 2.0
- Item - Contains default RSS 2.0 elements plus custom metadata elements. An example of a custom element would look like this:

- <date_issued>01-07-2007</date_issued>

Other enhancements

As said previously, RSS and ATOM are very flexible protocols, and various enhancements to it are already in use on the web. Three very useful technologies are Ping, Trackback and Authorization/Security. Because of time constraints, these three technologies have not been implemented in ATLAS, but considering that they are all supported by RSS and ATOM, it would be beneficial to add them to a working ATLAS system. A detailed explanation of the three technologies can be seen below.

- Adding “Ping” support for immediate content aggregation. Ping is a RSS service designed to send a notification to predetermined server(s), to let them know that the originating site has been updated with new content [19].
- Adding “TrackBack” support in order to verify whether the records transfer has completed successfully. TrackBack is a RSS service for peer-to-peer communication and notifications between two web sites.
- Adding security measures in order to ensure that the transfer of content and metadata between CMS and RMS takes place in a secure manner
 1. Authentication: HTTP Authentication is a basic authentication schema in which a user or application is required to input his credentials in the form of a user name and a password in order to gain access to a site. In HTTP Authentication, the user credentials are sent over the net in plain text, which makes interception by a third party possible. But if used together with HTTPS/SSL, the transmission becomes encrypted and thus secure.
 2. Confidentiality: HTTPS/SSL is used to ensure data confidentiality by encrypting the data stream between two communicating applications, and to authenticate the server, and client (optional). It can be used with the RSS protocol.
 3. Transfer verification: Using the TrackBack Ping functionality described previously, it becomes possible to verify that the Transport Package has been sent and received successfully. It should be noted that this does not guarantee that the content has not been altered or become garbled during transfer. The TrackBack Ping functionality does not guarantee the integrity of the contents of the Transport Package. For this, checksum or a similar kind of data verification would have to be added.

Support for other protocols

Whereas ATLAS is currently configured to work with RSS 2.0 and ATOM, it would be easy to add support for other protocols, as long as they meet the requirements mentioned under “Protocols” in section 6.1. An example of such a protocol would be a similar XML-based format for syndicating content and metadata. For such a protocol to be used with ATLAS, there are a few requirements: As with RSS 2.0 and ATOM, additional metadata fields must be supported. Whereas for example RSS uses a RSS/Channel/XML structure, the structure of a customized XML based protocol may be different. To use another protocol, this difference of structure would have to be defined in the CMS registration part of ATLAS [20].

6.2 Metadata schemas in ATLAS

It is difficult to generalize about what metadata needs to be exchanged between a CMS and a RMS. This depends on a wide number of factors such as system type, organization type,

organizational policy etc. What can be expected, however, is that the CMS and RMS will be using different metadata schemas. As an example to illustrate this, I have chosen two general metadata schemas: The “CMS metadata elements and guidelines” from Monash University to represent a CMS metadata schema, and “Requirements for Electronic Records Management Systems 2: Metadata Standard” from the UK National Archives. Table 1 shows an overview of the two metadata schemas [21][22].

Since the metadata schemas used here serve no other purpose, than to act as examples and to be used as schema representations when building a prototype of the ATLAS system, I have chosen to use only “Required/Required where applicable” fields and not include “Optional” fields. As it can be seen, some fields of the CMS metadata elements and guidelines are identical to Requirements for Electronic Records Management Systems (for example: Title). In those cases, no conversion is necessary. However, in some cases there may be differences in the data formats used between the two systems, such as the formats used to express Date or Identifier. In these cases, the values will have to be converted, using a crosswalk. Up to now, I have mainly been dealing with metadata relating to individual content. However, where this content belongs in the RMS classification schema (also referred to as fileplan) is also important. For example, the UK National Archives Requirements for Electronic Records Management Systems specifies 3 levels of metadata: Class, Folder, Record. Many CMS also support some form of Hierarchical structure, and there is nothing that prevents structural metadata to be converted and used, providing this metadata is explicitly expressed at the content level. Metadata that is implicitly inherited from a higher level can not be transferred, as it is not present at the content level.

Adding new metadata crosswalks

Once ATLAS has been set up and the metadata schemas to be used for the crosswalks have been decided, it is necessary to add these schemas to ATLAS. When development on ATLAS began, metadata schemas were simply hardcoded into the system, making changes to a particular schema difficult. The next step was to support the manual adding and editing of metadata fields, which made schema management easier.

However, in an organization with many different schemas in use, manual metadata maintenance in ATLAS becomes undesirable. This is partly because the tools in ATLAS for adding and changing fields are too simple, but also because schema reuse is difficult. There are two situations in particular where metadata crosswalk import functionality is needed. One of these is when ATLAS has first been setup, and the metadata crosswalks in use in the organization needs to be added. The other is whenever a new CMS needs to be added to ATLAS.

Creating metadata crosswalks

It becomes clear that there is a need for a simple way in which an archivist can handle crosswalks in ATLAS. Based on the problems described above, an ideal import function would enable schemas to be reusable internally in the organization, make them easy to import, editable using existing tools and finally, be human readable to ensure ease of data manipulation.

To that end, the import process in ATLAS has been based on the four technologies: XML, OWL, XSLT and JSON. XML has the advantages of being easy to work with and reusable throughout the organization, and OWL gives the added benefit of adding semantic functionality, such as the term validation via namespaces.

What is XML/OWL

Before looking at XML/OWL, it may be a good idea to start by explaining RDF. OWL and RDF are in many ways similar,

and OWL uses both the URIs for naming and the description framework for the Web that is provided by RDF.

RDF is short for Resource Description Framework, and is a language for representing information about resources in the World Wide Web. It is particularly used to represent metadata about Web resources, such as the page title, date etc. RDF is a machine-understandable language, which means that it is used for situations in where information needs to be processed by applications instead of only being displayed to humans. Just like XML is a common data exchange format, RDF gives a common framework for expressing information so it can be exchanged between applications. And just like XML it can be used with a number of processing tools and parsers.

In this format, the metadata can be parsed and used for metadata schemes in ATLAS. However, as mentioned earlier, the format used in ATLAS isn't XML/RDF but XML/OWL. OWL stands for Web Ontology Language, is a language for defining and instantiating Web ontologies. Ontology refers to the science of describing the kinds of entities in the world and how they are related. An OWL ontology may include descriptions of classes, properties and their instances.

OWL is a vocabulary extension of RDF, but OWL is a stronger language with greater machine interpretability than RDF. OWL also comes with a larger vocabulary and stronger syntax than RDF, making it well suited for use with ATLAS [25].

Creating XML/OWL crosswalks for ATLAS

The first step in creating the XML/OWL crosswalk, was to create a topic map of the crosswalk of CMS entities.

Using The Protégé Ontology Editor and Knowledge Acquisition System, the concepts and relationships between the individual metadata terms of the Drupal CMS were mapped to the RMS metadata terms using a graphical representation of the data (The resulting graphics file is too detailed to be shown in this paper, but can be obtained by contacting the author)

The graphical OWL visualization was then parsed and translated into the XML/OWL.

```
<rdf:li>
  <atlas:CmsElement rdf:about="http://www.nantoka.dk/#type">
    <atlas:element_path>type</atlas:element_path>
    <atlas:rms_element>Type.RecordType</atlas:rms_element>
    <dc:relation
rdf:resource="http://purl.org/dc/elements/1.1/#type"/>
  </atlas:CmsElement>
</rdf:li>
```

Figure 4: A single metadata element in the Drupal CMS XML/OWL

Importing crosswalks with XSLT and JSON

Once a crosswalk has been created in XML/OWL, the next step is importing it into ATLAS. Since the crosswalk is XML based, it is possible to use tools such as xpath¹ combined with a prepared script to extract the elements and values for import. However, there it is possible that the metadata schemas from

¹ Xpath is a tool that can be used to search and extract data from XML documents. It also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. [26]

different CMS to be imported will differ in length and structure. In such a case, the xpath script would have to be adapted to fit the new schema. Alternatively, the xpath script would have to be flexible enough to deal with different types of schema, adding greatly to the complexity. I have taken another approach, using the tools XSLT and JSON.

XSLT is a language for formatting XML documents through templates. In ATLAS, XSLT is used to transform an XML/OWL document into a JSON compatible format. To quote the JSON specification:

“JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999”. [27]

In other words, JSON is a lightweight format, far simpler than XML/OWL. JSON is also well supported by Ruby on Rails, on which ATLAS is built. This means that importing a string of CMS elements formatted in JSON is relatively simple, and can be performed with different crosswalks, even if the XML/OWL files are slightly different.

6.3 ATLAS Requirements

At its current stage, ATLAS has some requirements that need to be taken into account when implementing it.

The first of these has to do with the way metadata is handled. If the metadata provided by the CMS is very limited, or if it is very different from the one in use in the RMS, the archivist will have to fill out the missing data. This can be a very resource intensive task, if the number of transferred records is large. Furthermore, unlike the creator of the content, the archivist may not know enough about the content to add the correct metadata. One way to solve this problem is to extend the metadata schema used in the CMSs to contain the metadata fields used in the RMS.

The second point is that content is transferred to the archive “as is”. While retention periods tend to be short in corporate archives, some provisions may need to be made for ensuring future readability. In systems where long term data preservation is an absolute necessity, it is possible to imagine a conversion process taking place at the RMS Content layer before import (similar to the National Archives of Australia's XENA software) [28].

The third problem is more technical. I have tried to make my model flexible enough to work with any type of CMS and RMS, without having to any of these systems. However, some systems may not support the basic functions mentioned previously in this chapter. In those cases, this functionality will need to be added by hand, making implementation more costly.

These requirements aside, implementing ATLAS should be relatively straightforward for most corporations. This simplicity has been achieved by a number of different means. First of all, ATLAS has an advantage in the fact that its elements all lie within the control of the organization, making it possible to customize. It can work with off-the-shelf tools, which also means that it doesn't require changing or extending of the systems in use in the organization.

7. Implementation

Cost of Implementation

Implementation in this case means the cost associated with setting up a system for transferring records and their metadata from a CMS to a RMS. In reality, there are numerous factors that influence the cost of implementation, for example hardware or software cost, time spent on training, customization efforts, loss of productivity due to unfamiliarity

with new systems etc. However, it is possible to say that all other things being equal, the cost of purchasing and implementing new systems is more expensive than using existing systems, if there are no compelling reasons to do otherwise.

Provided that the organizations CMS and RMS fulfill the requirements for working with the three layered model mentioned previously in this paper, organizations can reap the benefits of automatic CMS to RMS archiving, without needing to exchange or modify their existing systems.

Price

The function of the the three layered model is to enable automatic CMS to RMS archiving. However, some CMS solutions on the market today offer integrated CMS and RMS functionality. Such systems (sometimes called Enterprise Content Management Systems (ECMS), are able to handle multiple types of content through all stages of the content lifecycle. In cases where an organization is already using an ECMS with records management functionality, there is no need to implement a system such as ATLAS.

However, it should be noted that, because of their size and complexity, ECMS are generally more expensive than smaller CMS and RMS solutions. A solution using several CMS connected to a single RMS via a system like ATLAS may prove to be cheaper.

Scalability

In a situation with one central ECMS to handle all aspects of content, including records management, what would happen if an organization were to find itself in a situation where it needed to manage a new type of content. For example, if an organization that had not done so previously would suddenly need to perform Digital Asset Management. If functionality managing this type of content were not available in the ECMS, the organization would need to either expand the ECMS (if possible and at a cost), or use a dedicated Digital Asset Management capable CMS. In the latter case, the organization would still need some way to transfer the new type of content to a RMS, and would in fact be able to benefit from a system like ATLAS.

Because of the built-in support for new CMS metadata schemas, scaling a system using ATLAS by adding an additional CMS should provide no real difficulties. This is provided the system is compatible with ATLAS in the first place. In other words, there would be no other cost to the organization, other than those associated with implementing the new CMS.

8. Evaluation

Before performing an evaluation, it is important to remember that this paper proposes two different things. The three layered model and ATLAS. In the evaluation below, I am talking about the three layered model unless specifically otherwise. The evaluation of ATLAS should be taken as an example of a way to implement the three layered model in practice.

Use

Regarding evaluation of the use of ATLAS, it should be mentioned that there are many different procedures for transferring content and metadata from a CMS to a RMS. I have chosen to evaluate ATLAS with two different scenarios, which I think represent the two opposite ends of the scale. One fully manual and one automated via export-import scripts. In reality, most organizations will probably use a solution that lies somewhere in the middle, ie. have some of the steps automated, while others remain manual.

However, before starting to analyze how ATLAS compares to other types of submission processes, it is necessary to

establish what records submission procedures are actually in use in organizations today. Unfortunately it is difficult to find reliable information on the internet about this area, since it pertains to internal organizational practices. However, the basic concept remains unchanged: If the CMS and RMS are incompatible, the only ways to transfer content is either manually, or through some sort of plug-in or script.

ATLAS compared to a fully manual records submission process

Since an automated process for records transfer would be less costly, there are still reason why an organization would maintain manual processes. As previously mentioned, one of the main reasons is system incompatibility. An example in point is the Royal Danish School of Library and Information Science. This organization uses Microsoft SharePoint as their main CMS, yet this system is incompatible with their RMS. The same problem was to be found at my company, where I worked as a records manager. The CMS (also SharePoint) was incompatible with the RMS (a Lotus Notes database).

The above incompatibility forces users to use manual a process for records transfer approaching the following steps:

1. Create final version of content in the CMS
2. Download content to local folder
3. Log-in to RMS
4. Upload content to RMS
5. Manually add necessary metadata
6. Save content as record

In this case, there are six steps to complete before the record is saved in the RMS. Furthermore, step 5 can be a very time consuming task, since all the administrative metadata from the CMS is lost in the process. Compare this with the ATLAS process below:

1. Create final version of content in the CMS

It must be noted that the above single step process depends on the metadata from the CMS to satisfy all the required RMS metadata fields. If this is not the case, missing metadata will have to be added manually.

Comparing ATLAS to a manual submission process, the number of steps is drastically reduced. When the user has created the final version of a record, it is automatically transferred, with metadata, to the organizational RMS.

ATLAS compared to semi-automated systems

Another way in which organizations transfer their files from a CMS to a RMS is through systems that automate parts of the transfer process, such as offering "bulk import" functionality in a RMS. An example of such a system is Oracle Records Database, which offers the ability of adding an entire file folder of files to the RMS.

In cases where this kind of functionality is not available in the system, organizations may have to write their own custom scripts or plugins to help with automation. An example of this can be found in a case study published by Information Management Journal in 2005:

"The bank examination staff was using a software package, developed in conjunction with several other federal bank regulators, to create and store examination work papers. The creation and preservation of complete, accurate, and trustworthy bank examiner work papers are paramount because bank ratings depend on examination results. The records manager asked the IT staff to develop a small set of computer code so that when a set of examiner work papers was saved by a bank examiner, a copy would be automatically sent to a folder controlled by the records manager" [29]

While being a very rudimentary example, it serves as an example of an organization having to develop a custom solution for automating export of records from a CMS.

So, how do ATLAS compare with such semi-automated processes? First of all, ATLAS covers the whole transfer process, from CMS to RMS. Secondly, ATLAS works with more than one CMS. In the bank case study above, the small set of computer code is designed to work with only one CMS, and is unlikely to work anywhere else. Thirdly, ATLAS supports transfer of CMS metadata and metadata conversion. Many customized solutions do not.

Other criteria for evaluation

There are a number of other areas where ATLAS can be said to have advantages, namely OWL functionality, crosswalk reuse and administration. Using OWL for its metadata crosswalks brings a number of benefits to ATLAS. First of all, by using OWL, organizations commit to keeping their metadata schemas in a universal format that has a common syntax XML/OWL and that machine understandable. This can be important in cases where organizations are using a large number of crosswalks, because it makes it easier to understand, update, and integrate legacy data when the platform is shared. Because of its semantic nature, using OWL also provides a common vocabulary of defined terms and the relationship between these terms. This makes searching the crosswalks easier, since the location of the terms and their interrelationship is defined in the ontology.

Finally, OWL also provides benefits to human users, since they can use OWL ontologies as a reference, for example by using the defined namespaces to lookup the meaning of metadata terms, or even visualize the data via visualization tools such as IsaViz. Using XML/OWL also makes it possible to parse crosswalks via OWL parsers to check for errors before upload to ATLAS.

As for crosswalk reuse, the XML/OWL metadata crosswalks in ATLAS provide an easy way to add a new CMS to ATLAS, namely templates. Once a template for a crosswalk has been created, a new crosswalk can be prepared in very little time by changing the XML tags pertaining to CMS.

Finally, using ATLAS give archivists or administrators a single interface for administering CMS schema and element information for all registered CMS. By using the Edit function in the CMS or Element management screen, archivists can perform simple management tasks such as adding extra metadata elements, changing the URL and so on.

8.1 Limitations of the three layered model

There are a number of limitations in three layered model as well. These have been described earlier in this paper. To summarize, there are a number of scenarios where a solution such as ATLAS would be difficult to implement: 1) If the CMS in question does not support RSS or a similar protocol for sending transport packages out of the box, this functionality will have to be added to the system, adding to the cost. 2) If the CMS uses only a very small set or metadata to manage content, or if the CMS metadata schema and RMS metadata schema are very different, a large amount of RMS metadata may have to be added manually by the archivist import. 3) If the content or metadata is "locked" in the CMS in such a way that it cannot easily be exported from the CMS. 4) If the RMS only supports adding content manually through a GUI (no support for bulk import).

9. Conclusion

In this paper, I have presented a new model for transferring records from CMS to an RMS. By using a lightweight three layered model, I have shown a way to reduce the complexity of integrating content management and records management

functionality compared to other models of digital archiving and harvesting of web content. This integration is of benefit to organizations, since it automates the records submission process, reducing the cost of users having to transfer content and metadata manually.

There is of course always a trade off between simplicity and functionality, and in some cases, the three layered model may turn out to be too limited. But I believe that the openness of the three layered model makes it possible to extend the basic functionality by adding extra functions such as content conversion. The fact that the three layered model doesn't require reprogramming of the organizations CMS or RMS, coupled with the fact that it is not tied to any one software solution makes it even more generally applicable.

Using the three layered model, I have developed the Automated Transfer Lightweight Archive System (ATLAS). ATLAS enables organizations to automatically transfer content and metadata from one or more CMS to an RMS. During the transport process, ATLAS performs automatic metadata conversion through metadata crosswalks. Metadata from the CMS is converted into a format which can be imported into the RMS, ensuring reuse of compatible metadata. By automating this process, the cost of making users manually enter metadata is greatly reduced.

ATLAS uses RSS 2.0 as its protocol for harvesting CMS content. Because of the almost universal support for RSS in current CMS, ATLAS can be used with most CMS out of the box with little or no need to perform alterations.

Finally, ATLAS offers support for uploading additional metadata crosswalks in XML/OWL. This upload process can be performed by archivists via the ATLAS GUI. This solution is more flexible than doing ad-hoc metadata conversion based on values hard coded into ATLAS because it makes it easier to add a new CMS to ATLAS. Using OWL for uploading new crosswalks can be useful for organizations who wish to store their data in a universal, machine understandable language, allowing better search capabilities and easier data integration.

By building ATLAS, I have shown that it was possible to automatically transfer content and metadata from an off the shelf CMS to a RMS. The metadata crosswalk functionality in ATLAS meant that CMS elements that could be mapped to RMS elements were translated and imported along with their respective content. The tested ATLAS solution was shown to be significantly less costly than a manual export/import process, and more flexible than a solution based on CMS specific or ad-hoc export/import scripts.

Finally, because ATLAS is built on open technologies such as RSS, XML, OWL, XSLT and JSON, it is possible to expand the current basic functionality by adding support for functionality such as RSS Trackback and Authentication.

REFERENCES

- [1]: Spotlight on Sarbanes-Oxley Rulemaking and Reports, U.S. Securities and Exchange Commission, <http://www.sec.gov/spotlight/sarbanes-oxley.htm>, Accessed 14-01-2008
- [2]: Robles, Marcel; Langemo, Mark. The Fundamentals of Records Management, Office Systems, 1999
- [3]: Mybrugh, Susan. Knowledge Management and Records Management: Is There a Difference, RIMR, September 1998
- [4]: XML to Work: Advantages of Content Management, XML.org, http://www.xml.org/xml/putting_xml_to_work.shtml, Accessed 14-01-2008

- [5]: Consultative Committee for space Data Systems. Reference Model for an Open Archival Information System (OAIS), Blue Book, 2002
- [6]: Boiko, Bob. Content Management Bible, 2nd Edition, Wiley Publishing, Inc., Indianapolis, 2005
- [7]: Internet Archive, The Internet Archive, <http://www.archive.org/index.php>, Accessed 14-01-2008
- [8]: Crosswalk: FGDC Content Standards for Digital Geospatial Metadata to USMARC, Alexandria Digital Library, <http://alexandria.sdc.ucsb.edu/public-documents/>, Accessed 14-01-2008
- [9]: Mary S. Woodley. Crosswalks The Path to Universal Access?, The J. Paul Getty Trust, http://www.getty.edu/research/conducting_research/standards/intrometadata/path.html, Accessed 14-01-2008
- [10]: WCSTools, Getdate, <http://tdc-www.harvard.edu/software/wcstools/getdate.html>, Accessed 14-01-2008
- [11]: Boiko, Bob. Content Management Bible, 2nd Edition, Wiley Publishing, Inc., Indianapolis, 2005
- [12]: New Plan Excel Realty Trust Integrates Enterprise Content Management, Oracle, <http://whitepapers.zdnet.co.uk/0,1000000652,260289614p,00.htm?dl=1>, Accessed 14-01-2008
- [13]: Consultative Committee for space Data Systems. Reference Model for an Open Archival Information System (OAIS), Blue Book, 2002
- [14]: Final Rule: Retention of Records Relevant to Audits and Reviews, U.S. Securities and Exchange Commission, <http://www.sec.gov/rules/final/33-8180.htm>, Accessed 14-01-2008
- [15]: Open Archives Initiative Protocol for Metadata Harvesting, v. 1.1. 2001, Open Archives, http://www.openarchives.org/OAI_protocol/openarchivesprotocol.html, Accessed 14-01-2008
- [16]: Lagoze, Carl; Van de Sompel, Herbert. The OAI: Building a low-barrier interoperability framework, www.openarchives.org/documents/jcdl2001-oai.pdf, Accessed 14-01-2008
- [17]: RSS 2.0 Specification, RSS Advisory Board, 12 August, 2006, <http://www.rssboard.org/rss-specification>, Accessed 14-01-2008
- [18]: Recasting the Concept of Podcasting, TDG Research, <http://news.digitaltrends.com/talkback109.html>, Accessed 14-01-2008
- [19]: Understanding Blog and Ping, Blog Herald, August 16, 2005, <http://www.blogherald.com/2005/08/16/understanding-blog-and-ping/>, Accessed 14-01-2008
- [20]: TrackBack Technical Specification, Six Apart: Developer Documentation, <http://www.atomenabled.org/developers/syndication/>, Accessed 14-01-2008
- [21]: CMS metadata elements and guidelines, Monash University, <http://www.lib.monash.edu.au/metadata/cms-metadata.html>, Accessed 14-01-2008
- [22]: Public Record Office. Requirements for Electronic Records Management Systems, Crown, 2002 revision, final version
- [23]: RDF and OWL Recommendations, World Wide Web Consortium, <http://www.w3.org/2004/01/sws-pressrelease>, Accessed 14-01-2008
- [24]: RDF Primer - W3C Recommendation 10 February 2004, World Wide Web Consortium, <http://www.w3.org/TR/REC-rdf-syntax/>, Accessed 14-01-2008
- [25]: OWL Web Ontology Language Guide, World Wide Web Consortium, <http://www.w3.org/TR/owl-guide/>, Accessed 14-01-2008
- [26]: XML Path Language (XPath), World Wide Web Consortium, <http://www.w3.org/TR/xpath>, Accessed 14-01-2008
- [27]: Introducing JSON, JSON, <http://www.json.org/>, Accessed 14-01-2008
- [28]: Tools for digital preservation, National Archives of Australia, <http://www.naa.gov.au/records-management/secure-and-store/e-preservation/at-naa/software.aspx>, Accessed 14-01-2008
- [29]: Electronic records management on a shoestring: Three case studies, Entrepreneur.com, <http://www.entrepreneur.com/tradejournals/article/127433398.html>, Accessed 14-01-2008