An Overview Over Content Management System Integration Approaches

An Architecture Perspective on Current Practice

Hans-Werner Sehring
Namics
Hamburg, Germany
e-mail: hans-werner.sehring@namics.com

Abstract—**In practice, content management systems are in widespread use for the management of web sites, for intranet solutions, and for the publication of a range of documents created from diverse content. An emerging class of multimedia databases is digital asset management systems that specialize in the management of unstructured content. Despite the market for content management products aiming at integrated solutions that cover most content management aspects, there is a trend to augment content management systems with systems that offer dedicated functionality for specific content management tasks. In practice, there is particular interest in systems incorporating both a content management system and a digital asset management system. All integration forms exhibit individual strengths and weaknesses, achieved with differing implementation effort. The choice of the adequate integration architecture, therefore, depends on many factors and considerations that are discussed in this paper.**

Keywords-content management; digital asset management; software architecture; solution architecture; systems integration.

I. INTRODUCTION

Content Management Systems (CMSs) are in widespread use today for the maintenance of web sites by content producers and editors. Typical CMSs aim to manage both structured content (often in the form of hierarchies or graphs of content objects) and unstructured content, namely binary data that is shipped as some media file of a certain standard format (like, e.g., images and videos in different formats).

In practice, CMSs host elaborate processes that deal with structured content while offering only very basic functionality for unstructured content. CMS customers have an increasing demand for additional functionality for the treatment of binary multimedia content [1].

Consequently, there is a current trend to augment CMS installations with a multimedia database of the newly emerged class of *Digital Asset Management systems* (DAMs).

Both CMSs and DAMs provide a complete feature set for the management and distribution of content, the major difference being the form of content they specialize in. Since both CMSs and DAMs are designed to manage content and publish it on the web, their integration therefore is not obvious. In fact, depending on the particular requirements of a web site, different integration forms are suitable, each providing its own advantages and drawbacks.

In this paper, we discuss integration approaches for systems consisting of a CMS and a DAM. All approaches considered are derived from actual scenarios found in commercial projects. They all assume the CMS to deliver web pages and the DAM to contribute embedded multimedia documents [2]. The integration approaches differ in the point within the content lifecycle at which the DAM contributes.

The remainder of this paper is organized as follows: In Section II, we discuss the characteristics and functionality of CMSs and DAMs. In Section III, we review the lifecycle of content and digital assets, respectively, in typical CMS and DAM implementations. Section IV constitutes the main part of this paper. It presents the integration forms that correspond to certain lifecycle states. Each integration form requires some adaptations to the CMS or the DAM. These additions are discussed in Section V. Section VI presents a slight variation of the integration scenarios in the way that instead of plain assets a produced document is handed over to the CMS. The paper concludes with a summary and outlook in Section VII.

II. CONTRIBUTING SYSTEMS AND THEIR FUNCTIONALITY

With CMSs and DAMs there are two classes of systems that deal with the editing of content and shipping of content.

Both contain editing facilities including workflows and quality assurance processes. Both offer rendering and playout functionality, usually targeted at specific usage scenarios. These scenarios differ between software products (performance, editing of unique documents vs. management of uniform mass content, etc.).

As the names indicate, the systems differ in the kind of entities they deal with. CMSs focus on the management of structured content and on publication of documents that are created from compositions of pieces of content. DAMs deal with unstructured content that is managed, transformed, and published on a binary level.

Consequently, CMSs and DAMs address similar use cases, but they put a different focus on the functionalities as discussed in the subsequent subsections.

A. *Content Management Systems*

CMSs provide their service as follows (see also [3]).

1) Content creation: CMSs offer tools for manual creation of content by editors and for the import of content from external sources, be it from files, from feeds, or by means of content syndication.

2) Content editing: Part of a CMS is an editor tool that is used to manipulate content, to control its life cycle (see

Section III), and to preview renderings of content. Content manipulations include adding value to content, the maintenance of description data, and the addition of layout hints and other channel-specific settings, e.g., URLs for the publication of content in the form of world wide web resources. Editing tools can be form-based with a separate preview or in-document, in which case the editor manipulates documents, and manipulations are mapped to the corresponding content. Often there are workflows to control the editing processes.

3) Quality assurance: Quality assurance for content consists of approval and publication, although in some CMS products these two activities are one. Approval marks content as being suitable for publication. Publication finally makes it available to the target audience – in the form of rendered documents. Quality assurance should be embedded in the CMSs workflows.

4) Rendering: Rendering is the process of creating documents from content. Structured content typically is rendered by mapping content structures to document layouts. The ability to manipulate binary content is limited compared to that of a DAM with matching capabilities. CMSs offer general functionality on media content suited for a particular publication channel, e.g., for the web. This particular case includes rendering of images for adaptive design, e.g., to resize them for specific channels or to apply device-specific format conversions.

5) Playout: The shipping of rendered documents, called delivery or playout, is not necessarily a core functionality of a CMS. But since playout usually is tightly coupled with rendering, CMS products include a playout component. Some CMSs target high performance output, sometimes being integrated with Content Delivery Networks (CDNs).

B. Digital Asset Management Systems

A DAM's functionality includes the following [4].

1) Asset Creation: Assets are created in a DAM as content is in a CMS, manually or in automated processes. Manual creation is typically accomplished by means of an external authoring tool. Its output is uploaded to the DAM.

2) Asset Editing: Editing is typically restricted to the maintenance of structured information (descriptive data, e.g., defining time code information in moving image, legal information, provenance information, etc. [5]). Binary manipulations are performed by authoring tools. Editing may take place in workflows [6].

3) Quality assurance: DAMs have an approval process like the one of CMSs. Workflows for quality assurance can typically be customized.

4) Rendering: The rendering of digital assets consists of format conversions, media manipulations, and generating multimedia documents from multiple assets. Transcoding particular video formats for different browsers or mobile platforms is a typical manipulation task. Manipulations include image manipulation, e.g., scaling of images for adaptive design, inserting logos in photos, watermarking of documents, etc. An example for on-the-fly document generation is assembling a video from moving image and sound for multilanguage videos. Whole hypermedia documents can theoretically be created this way. Another example is the addition of descriptive data to multimedia assets as meta data, e.g., Exif data.

5) Playout: DAMs typically can deliver assets, at least by shipping online to the web or offline by creating files, e.g., for print. Some DAMs offer more sophisticated playout functionality, e.g., reliable delivery, at-most-once delivery, exactly-once-delivery, or digital rights management. DAMs specialized in video management offer a playout based on QoS parameters. In particular, they measure network latency during video transmission to be able to sacrifice image quality in favor of synchronicity if needed [7].

III. CONTENT AND DIGITAL ASSET LIFECYCLES

Both content objects managed by a CMS and assets managed by a DAM have a lifecycle. In most products, these lifecycles are explicitly represented by states of the objects. Figure 1 illustrates the states and possible state changes as described below.

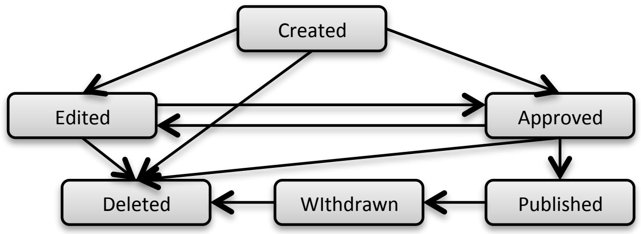

Figure 1. Lifecycle states of content objects.

The content object lifecycle starts with content objects being created. This can happen manually or by importing external content, e.g., from files or news feeds.

Subsequent editing adds value to content. Changes affect the actual content or descriptive information that is also stored in content objects. In particular, editing may include linking content objects to each other in order to create multimedia documents from the resulting object graphs.

Quality assurance for content is reflected in a dedicated approval step that marks content as being suitable for publication. Such content is, depending on the CMS product, either directly available for rendering and shipping or it constitutes a candidate for a final publication step. In the course of this paper we draw no distinction between publication and approval.

An approved object that is edited becomes unapproved. Typically CMSs support versioning of content and this way allow the approved version to be online and a newer version to be edited.

In many states, a content object can be deleted.

Assets, being a different form of content, have a similar lifecycle. They are initially created inside a DAM, be it by import from external sources or by original authoring and

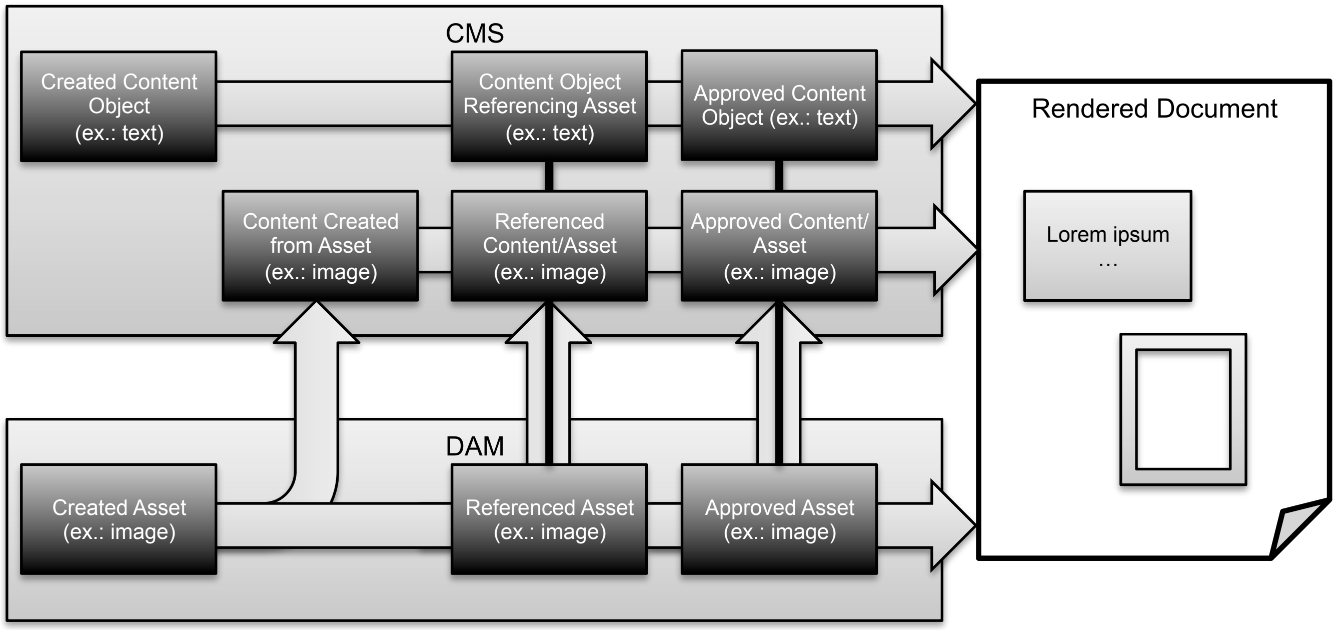
CMS
Created Content Object (ex.: text)
Content Object Referencing Asset (ex.: text)
Approved Content Object (ex.: text)
Content Created from Asset (ex.: image)
Referenced Content/Asset (ex.: image)
Approved Content/ Asset (ex.: image)
DAM
Created Asset (ex.: image)
Referenced Asset (ex.: image)
Approved Asset (ex.: image)
Rendered Document
Lorem ipsum …

Figure 2. Example content and asset lifecycle and relationships.

storing the results inside the DAM. Editing assets is not a primary use case of a DAM [8], so we omit asset modifications here. DAMs support quality assurance by an approval process, though, similar to that found in CMSs.

IV. Time of Asset Integration

Even if the management of structured and that of unstructured content are separated utilizing a CMS and a DAM, respectively, content and assets need to be combined in published documents.

There are various integration scenarios to achieve this kind of separation of concerns. For a concrete system the integration approach should be chosen based on the requirements that the system needs to fulfill and the implementation effort. On that basis, the most beneficial approach can be chosen.

Each integration form has its specific advantages and disadvantages and addresses a different set of requirements. The subsections of this section discuss one approach each.

The subsequent Section V discusses the implementation effort of each integrated solution.

For the integration scenarios we only consider the case of a CMS being used to prepare content and to define how to render documents. This is the particular strength of a CMS that cannot be substituted by a DAM. Therefore, the CMS will always be in lead when considering the overall document publication process.

The approaches thus differ in the point in time at which an asset is integrated into the CMS. **Figure 2** illustrates the scenarios covered in this paper by different content flows.

A. Integrating Assets at Playout Time

The integration at playout time makes full use of the DAM's functionality with respect to rendering and playout. Documents are created from both content and assets at the latest point in time possible. This way, it is the loosest integration form that happens at the point of document assembly. The equivalent in an information system is the presentation layer.

Though this frontend integration makes this approach the most volatile one, it is often preferred in practice due to its comparably low implementation costs and due to the fact that all of the DAM's functionality is being used.

A CMS's editor tool allows content objects to be related to each other. Such relationships are required either to be able to link documents or to define content structures that lead to documents composed of various content objects. Figure 2 uses the example of an image related to text. This integration scenario – as well as all the other ones discussed in the course of this paper except for the integration at creation time – requires an extension of the CMS's editor tool with a search in the accompanying DAM. At the same time the search functionality of the DAM is required to be exposed to the CMS.

For integration at playout time the CMS stores proxy content (as asset references) only at editing time. Such proxy content represents an asset from the DAM. It is created when an asset reference is defined using the editor tool.

The external references from proxy content to the asset it represents require the DAM to provide stable external asset IDs or addresses.

The CMS renders proxy content objects as references to the according assets residing inside the DAM that delivers them directly into the documents.

After creation of proxy content the CMS needs to receive events concerning the asset's lifecycle. A referenced asset might become unavailable for publication later on due to disapproval or deletion from the DAM.

There is no general way to prevent possible runtime errors due to assets that have been deleted or ones that have otherwise become inaccessible. Depending on publication strategies all content referencing such an asset may become inaccessible, as well as (transitively) all content referring to such content. In other cases, it might be possible to remove such references but leave the rest of the content intact.

In the case of web content management this scenario requires the DAM to be exposed to the Internet in order to be able to deliver the assets for inclusion into documents.

B. *Integrating Assets at Render Time*

Like most of the integration scenarios this one requires (see previous subsection): an extension of the CMS's editor with a search in the accompanying DAM, capabilities to manage asset references in order to relate assets to content, and means to deal with the fact that asset and content life cycles cannot be synchronized in a generic way.

During rendering, references are resolved. Assets are transferred to the CMS and stored at least in the public stage. The benefit of this step is increased independence from the asset lifecycle from this point on: asset deletion no longer leads to inconsistent publications out of the CMS. Nevertheless, disapproval of an asset does not automatically lead to withdrawal of corresponding and referring content.

The problem with unavailable assets exists as in the preceding case. Yet it does not occur at playout time, but instead at rendering time. This makes no difference in most contemporary CMSs. In offline CMSs that render documents in advance, this can be beneficial, though.

C. *Integration Assets at Approval / Publication Time*

This integration scenario is much like the preceding ones, only that it integrates assets even earlier in the asset/content lifecycle, namely during approval or publishing.

Typically, content is published in a transitive way. E.g., when an article is published, all related images need to be published in the same step as well, or otherwise the publication of the article will fail.

This integration scenario is based on an extension of the CMS's approval process in a way that assets are retrieved from the DAM and stored as content in the CMS during the process (based on proxies created at editing time), at least in the public stage. This scenario is based on the assumption that it is insufficient to apply quality assurance to the proxies alone because of asynchronous asset modifications in the DAM. Instead, the assets' approval state is checked as part of the approval process of the CMS.

In contrast to the preceding scenarios, the CMS is leveraged from having to consider unavailable assets at playout time. Still, the decoupled life cycles of asset and corresponding content need to be dealt with. To this end, there either needs to be a synchronization of asset and content state based on notifications as discussed before, or the CMS neglects the approval state in the DAM and maintains the state on the basis of content objects only.

In this integration scenario, as opposed to the preceding ones, the CMS's publication, rendering, and playout capabilities are used for digital assets. Section V.B discusses the resulting implications. The DAM's playout functionality (see Section II.B) will not be utilized.

D. *Integration Assets at Editing Time*

Assets can be added to the CMS at editing time, e.g., when a reference to an asset is added to some content. This requires an extension of the CMS's editor with (a) search in the accompanying DAM like in the cases above and (b) on-the-fly content creation from selected assets.

If assets are integrated in the CMS before approval they need to be monitored for subsequent changes. To this end, there needs to be synchronization once content has been created from an asset. This synchronization may be eager (on every asset change) or lazy (on demand, e.g., at playout time).

With integration at approval time and before, rendering and playout are performed by the CMS (s.a.).

E. *Integrating Assets at Asset Creation Time*

The earliest possible integration of assets is at the time of their creation: assets are added to the CMS as soon as they are created in the DAM.

This scenario only makes sense if the DAM is also used in processes other than document production through a CMS. Otherwise there would be no need for a DAM at all. When assets still have an independent lifecycle inside the DAM then the integration requires continuous synchronization. This synchronization is performed eagerly in order to provide assets as content for selection within a CMS. There is no need for an extended editor that allows searching the DAM since assets can directly be found in the content base.

In this scenario, nearly all DAM functionality is neglected in favor of the corresponding CMS functions. As in the above scenarios quality assurance is controlled by the CMS, and rendering and playout are carried out solely by it.

V. REQUIRED SYSTEM ADAPTATIONS

In order to implement the integration of a CMS with a DAM in one of the forms presented in the preceding section, some extensions or adaptations to the software products are required. Table I gives an overview of required adaptations and attributes them to the integration scenarios.

A. *Added Functionality*

The scenarios that rely on a continuous synchronization of assets and corresponding content objects are typically implemented through notifications by events, e.g., the event of an asset having been modified. In these scenarios the DAM needs to be an event source and the CMS an event subscriber. The DAM will produce events and transmit them to subscribers. The CMS registers for such events and to interpret them. When this functionality is not found in the CMS (which usually is the case), there needs to be an external software component that listens to such events and then triggers some actions inside the CMS. To this end the CMS needs to provide an externally usable API.

In order to relate events to content created from assets, the DAM has to provide stable IDs or addresses (like, e.g., URLs) of assets. This is particularly important due to the fact that assets are long-lived.

Most events are related to specific revisions of assets. For those events subscribers need IDs that reference asset

TABLE I. CHANGES TO SOFTWARE PRODUCTS DEPENDING ON ASSET INTEGRATION TIME

Aspects	Form of Integration				
	Creation time	*Editing time*	*Approval time*	*Render time*	*Never*
Changes to CMS	• subscribe to and listen to events (from DAM) or expose public API; create content on asset creation or modification	• media selection dialog changed to query DAM • on-the-fly content creation upon asset utilization (linking) • subscribe to and listen to events (from DAM) or expose public API; modify content on asset modification	• media selection dialog changed to query DAM • surrogate objects for assets • on-the-fly content creation on public stage upon asset (proxy) approval • check of asset's approval state upon asset proxy approval	• media selection dialog changed to query DAM • surrogate objects for assets • on-the-fly content creation on public stage upon asset (proxy) rendering	• media selection dialog changed to query DAM • surrogate objects for assets
Changes to DAM	• event source for CMS • stable external IDs (to relate assets in events)	• query interface for CMS • event source for CMS • stable external IDs (to relate assets in events)	• stable IDs/addresses • query interface for CMS • interface to query approval state from CMS	• stable IDs/addresses • query interface for CMS • event source for CMS	• stable IDs/addresses • query interface for CMS
Unused CMS functionality			• quality assurance	• quality assurance • rendering (assets)	• quality assurance • rendering (assets) • playout (assets)
Unused DAM functionality	• rendering • playout	• rendering • playout	• rendering • playout	• playout	

revisions, not assets in general. For an example of IDs fulfilling this requirement see the CMIS object IDs [9].

As described in the preceding section, some integration scenarios rely on an asset selection dialog integrated into the CMS's editing tool. Usually, such a dialog exists, but is used to select multimedia content from the CMS itself. This dialog has to be extended in a way that allows picking assets from the DAM that have not previously been imported into the CMS. Such a dialog must furthermore be backed by functionality to create content from the chosen asset, either with a copy of the content or with a link to the asset. In order for the asset selection to work the DAM has to offer search functionality to the CMS (editor). The search result contains, depending on the scenario, the asset data or the asset ID or address.

B. *Unused Functionality of the Software Products*

There exists functionality that is provided both by a CMS and a DAM. In an integrated system the corresponding functions of one the systems may not be used. From an architectural point of view, this makes no change. But certain strengths and weaknesses of the products might not be considered in an optimal way in particular integration scenarios.

In those integration scenarios where the CMS handles references to assets in the DAM only, the quality assurance measures, usually some approval process, of the CMS are not in effect for assets. Approving a content object just makes a statement about a version of the corresponding asset at approval time, but assets may change without the handles inside the CMS being altered.

The aforementioned event-based synchronization can be used to monitor the approval state of assets and to adjust the approval state of the corresponding content objects. But considering the whole asset lifecycle there are situations that cannot be handled. The most drastic example is a valid asset that is (rightfully) referenced by published content. If now the asset is deleted then the CMS notices the state change. But it cannot decide whether to keep the image reference (thus rendering documents with missing images), whether to remove the images reference from all content objects (thus automatically altering the content; an operation that is usually unwanted in CMSs), or whether to disapprove all content objects containing the image reference (an operation that has to be applied recursively and can thus have unexpected effects).

If integration of a CMS and a DAM takes place in a way that assets are copied to the CMS before playout time, the rendering and possibly playout functionality of the DAM will not be utilized. This is a major drawback of those integration scenarios since these are about the most powerful contributions of a DAM. A CMS typically offers very limited rendering functionality for multimedia content, if any (see Section II.A). In the subsequent Section VI, we discuss an integration approach that allows to use more of a DAM's rendering functionality. Playout with QoS parameters is usually not provided by a CMS, but by some DAMs.

If integration of a CMS and a DAM takes place at a point in the asset lifecycle later than content editing, the rendering and possibly playout functionality of the CMS is not used for content originating from assets. As pointed out above, the corresponding functions of a DAM are typically more powerful that those of the CMS (see Section II.B). But there are some things to consider in specific scenarios.

The rendering of assets often is influenced by context-specific parameters of the publication channel at hand. For adaptive web design, for example, images are scaled to the actual screen size of the device posing a request, videos are transcoded to suitable formats, etc. In addition, some CMS installations allow editors to define the image formats used in particular situations, e.g., renderings in certain contexts. This cannot be achieved as easily when the DAM has the duty of rendering assets.

With respect to playout a CMS does not provide the media-specific functionality found in a DAM, in particular there is no quality-controlled adaptive playout. On the other hand, the CMS uses a playout infrastructure consisting of sophisticated caching, inclusion of content delivery networks, etc. This infrastructure has partly to be made available to them DAM.

VI. ADVANCED SCENARIO: ASSET SHIPPING TO CMS

From an editing viewpoint the integration at the time of asset creation or editing time is the most beneficial. To allow more of a DAM's rendering functionality to come into play in such an integration scenario, a variation of the corresponding integration approach can be taken.

In the preceding section we assumed the systems to pass "raw" content to the other, limiting the DAM to a multimedia database. Alternatively the synchronization of asset content can be considered a logical playout step from the DAM with the CMS being the receiver of rendered documents.

Though this variant does not help for playout (QoS parameters, etc.), it allows the integrated system participating in the DAM's functionality to render multimedia content (see Section II.B).

Particular attention has to be put on the interplay of the DAM's and the CMS's media manipulation functionality. E.g., a graphic would be stored in raw format inside the DAM. It provides a rendered version to the CMS, e.g., in a predefined format and resolution. During the shipping of the content from within the CMS this will in turn prepare the graphics data by scaling it for the usage at hand (full screen version, smaller embedded version, high resolution print version). The concatenation of the manipulation functions may lead to quality losses compared with a one-step rendering through the DAM's rendering functions.

In cases where there is no interference between the DAM's and the CMS's rendering of assets, the concatenation allows combining the quality of renditions provided by a DAM and the control over renditions by a CMS editor.

VII. SUMMARY AND OUTLOOK

The paper closes with a summary and an outlook.

A. *Summary*

This paper presents various forms of integration of a CMS and a DAM. If the CMS is in lead regarding the overall content management process then the main difference between the integration forms is the point in the asset lifecycle at which an asset is introduced in the CMS.

All integration forms exhibit individual strengths and weaknesses, achieved with differing implementation effort. The choice of a suitable integration form, therefore, depends on many factors and considerations discussed in this paper.

B. *Outlook*

For integrated solutions – like a CMS combined with a DAM in this case – we would like to see a repository of typical requirement/solution patterns.

The discussion in this paper shows that many decisions rely on the particular properties of the software products used. The solution scenarios should therefore be refined to consider actual software products with their individual capabilities to be of increased value in practical applications.

Furthermore, some decisions have to be made on the basis of more concrete requirements: the integration approach in general, but also implementation details like, e.g., the way how to handle concurrent asset modifications in the DAM and in the CMS. A comprehensive catalog containing more refined use cases and blueprints for typical solutions is required in practice.

Future work will try to extend the considerations to more general integration scenarios in the field. A quite prominent example is product information management fulfilled by, e.g., a CMS in cooperation with catalog management or a CMS combined with a shop solution.

ACKNOWLEDGMENT

The author thanks his employer, Namics, for the opportunity to follow his scientific ambitions by publishing some of his thoughts. In particular since these relate to and extend commercial activities.

Fruitful discussions with numerous colleagues (current and former), business partners, and customers led to the insights presented in this paper. My thanks go to all of them.

My thanks also go to the anonymous reviewers for the constructive hints that helped to improve this paper.

REFERENCES

[1] Ovum, Making the case for digital asset management in retail: Using technology to manage digital assets effectively. Whitepaper, August 2015.

[2] A. Saarkar, Digital Asset Management. Whitepaper, Cognizant Technology Solutions, 2001.

[3] S. King, "Web content management", in Computer Technology Review. Los Angeles, vol. 22, issue 11, p. 9, 2002.

[4] D. Austerberry, Digital Asset Management: How to Realise the Value of Video and Image Libraries. Amsterdam, Boston: Focal Press, an imprint of Elsevier Ltd., 2004.

[5] Y.-M. Kim et al., "Enterprise Digital Asset Management System Pilot: Lessons Learned", in Information Technology and Libraries, John Webb, Ed. vol. 26, no. 4, 2007.

[6] T. Blanke, "Digital Asset Ecosystems: Rethinking crowds and cloud", Chandos Publishing, 2014.

[7] H. Thimm and W. Klas, "Playout Management in Multimedia Database Systems", in Multimedia Database Systems, K. C. Nwosu, B. Thuraisingham, and P. B. Berra, Eds. Springer US, pp. 318-376, 1996.

[8] C. D. Humphrey, T. T. Tollefson, and J. D. Kriet, "Digital Asset Management", in Facial Plastic Surgery Clinics of North America, vol. 18, no. 2, pp. 335-340, 2010.

[9] Content Management Interoperability Services (CMIS) Version 1.1. 23 May 2013. OASIS Standard. [online]. Available from: http://docs.oasis-open.org/cmis/CMIS/v1.1/os/CMIS-v1.1-os.html