# Integrating DAM in the enterprise architecture

**Peter Hausken**
is CIO of J.W.Cappelens Forlag AS, a major Norwegian Publisher with a diverse range of publications, book clubs and a distribution central. Cappelen has worked with transforming the IT architecture to make it a modern and agile platform for the future. Before joining Cappelen in 1995, Peter worked for Hewlett-Packard, University of Oslo and the national academic network, UNINETT. He has a Masters Degree in Computer Science from the University of Oslo.

**Paul Heisholt**
works as a system developer at J.W.Cappelens Forlag AS. Before joining Cappelen in 2001 he worked for the National Norwegian News Agency (NTB). First as a journalist on the foreign desk. Later he became involved in editorial news system integration and development at the news agency. He has a Bachelors Degree in Political Science from the University of Oslo.

**Keywords**: Integration, Digital Asset Management (DAM), Service-Oriented Architecture (SOA), Web Services

**Abstract**: The Digital Asset Management system should not be just another system, but an integral part of the IT architecture. This paper outlines why and how a publishing company did this.

## Introduction

Cappelen started investigating the area of DAM in 2000. It soon became clear that there were many areas where digital assets were involved, and that storage and retrieval of these assets depended upon how the employees handled them. Multiple copies, different versions and even multiple acquisitions of the same asset manifested the need for a common solution. This is a situation known to most companies struggling to get their house in order. The digital revolution has made it easy to create and store all kinds of digital files, but yet so hard to find and retrieve when needed.

## Fundamental principals

We made a list of some basic requirements as we started our quest for a solution. These have later become very fundamental to how we implement DAM in our organization.

- A Digital Asset Management system had to be *flexible* and *useable* in different business processes.
- The DAM solution had to be so *easy to handle* that everyone could use it with minimal training.
- DAM had to be *integrated with other systems* and not be an isolated system

## Flexibility

Our DAM system needed to be tailored to handle a variety of tasks. Among them to store manuscripts and illustrations in their various stages throughout the production cycle, and in addition to store covers, excerpts and author portraits. Also publishing book club magazines on paper and web may require a different approach from storing audio books.

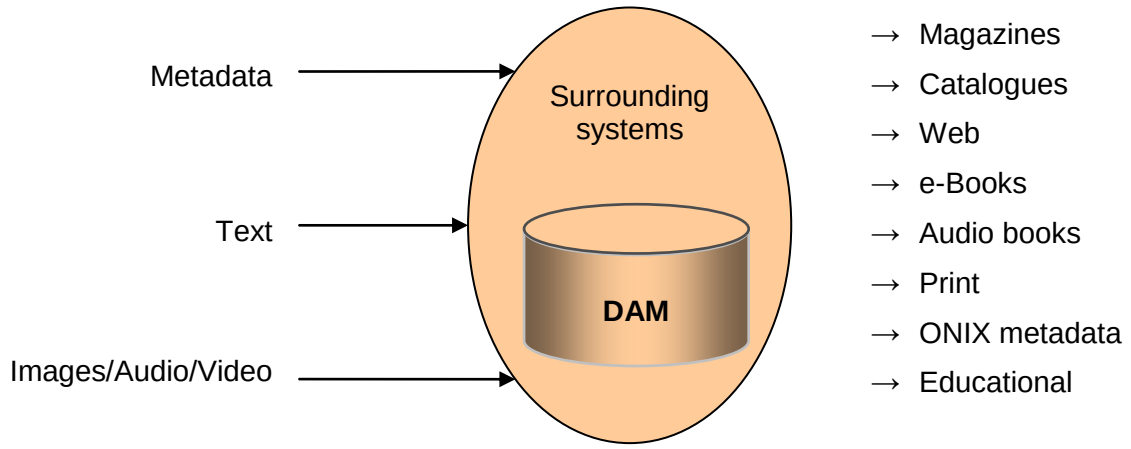
One can argue that not all of this is Digital Assets in its own right. But we were determined to have only one single repository for both real assets and more temporary content.

**Easy to learn and easy to use**
The standard DAM systems are naturally oriented towards their main purpose of organizing assets in a way that makes it easy to retrieve them and their corresponding metadata. This often makes the task of storing various assets correctly a specialized one. If things are not stored properly according to standards, it will end up either as an over simplistic system or just a plain mess. Both can make it hard to retrieve the right assets when needed.

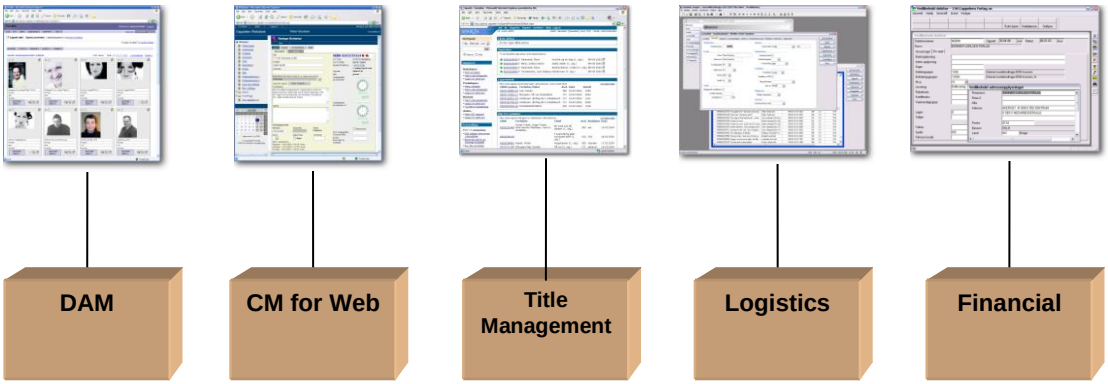
Norway is a high cost country with only 4.3 million inhabitants, and a language of its own. Hence Norwegian book publishers face a rather small market. With high wages and no cheap labor pool, it is too expensive to have dedicated people handle all the different asset types in and out of a DAM system. It was therefore necessary to create a system so user-friendly that the professionals could handle it themselves. For an employee in editorial, marketing or production, storing assets is often a very minor task. So Cappelen decided to make Digital Asset Management an integral part of the systems that are crucial to their everyday work.

**Integration to other systems**
When analyzing the work flow of registering new titles, we discovered that the same data was entered into multiple systems, which gave ample room for errors. This was no surprise, but rather a result of how various systems had been acquired for different purposes without the proper technology to integrate them online in a secure fashion.

**The Goal**
So what we needed was a system that was easy to learn, easy to use, and could handle Digital Assets as an integral part of the work flow. At the same time it had to be a one-stop-shop for all book related information.

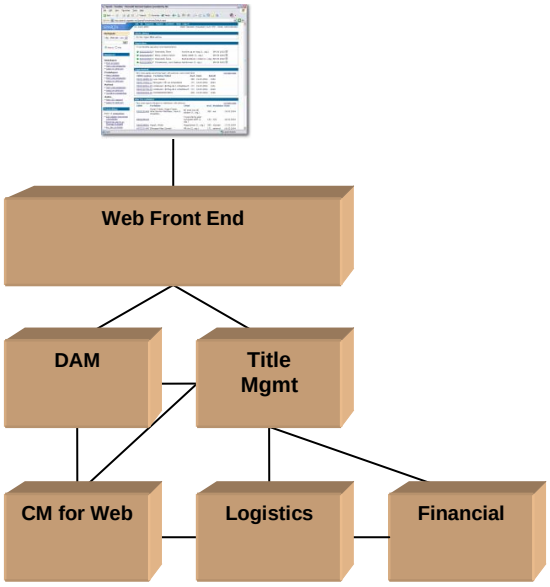



**Editorial system**
And guess what? That system did not exist as a finished product that could be bought and implemented as a quick project. We could buy the Digital Asset Management system, but found that it would require substantial development efforts to tailor it to our needs. And even if we tailored the DAM User Interface and made some of the integration, it would still be seen as a new system to handle a new task.




Our vision was to create a system that would appear as one single application to the professional users in the editorial, marketing and production departments. Not yet another application in addition to all the other existing applications. In fact we wanted to *reduce*, not *increase*, the number of applications.

To resolve the dilemma we decided to replace the user interface of the DAM system all together, and just use the APIs (Application Program Interface) provided by the DAM system to create our own user interface.




This approach also enabled us to remove functionality for handling various information about the books from the antiquated Logistical System and a couple of other minor systems and place them all in a separate module for Title Management and Production Planning.

This would give us a complete Product Management System for all information about the books. Like authors, formats, production plans, marketing plans, relationships between various products etc. This is something that varies from business area to business area, because the products are so diverse. Standard systems tend to deal with just part of the information you really want about your products. So by making product information management flexible and independent of any other system has later turned out to be of great value in other projects. It saved us big money when we later replaced the Logistical System. The authoritative source for all data about a book is the Title Management System. Other systems, like Logistics and Financial just have copies of the few data elements they need.

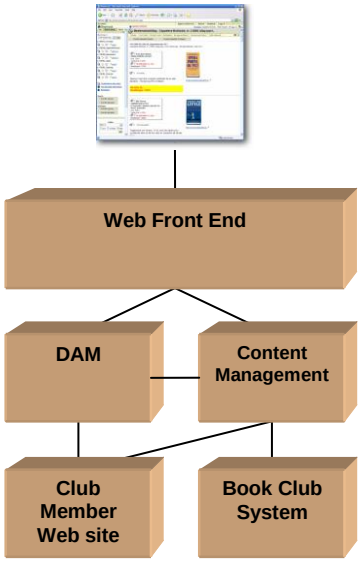
Two applications - a Central Book Archive and a Production Management System - were integrated into our new system, and the old applications were taken out of production.

The Content Management System for publishing to the company's web sites is now fed directly with data from DAM and Title Management. This makes it easier to assure that the information is delivered in time.

Book catalogs and other printed marketing material are also extracted from the system, as it has become a one-stop-shop for all product information.

**Book Clubs**

For the book clubs we had similar requirements to make the world appear easier for the users. At the same time we had to accomplish more with less people by using technology.




The aim was to create a system that could produce book club magazines and feed content to the book club web sites from the same data. And it had to be simple and as user-friendly as possible. In other words, we made a multi-channel publishing system with our own user interface, and integrated the DAM system as a module within that.

The Book Club System runs on a mainframe computer with an old-fashioned terminal interface. Many younger employees are very unfamiliar with the 3270-type interface, and the system has a long and hard learning curve for people who have grown up with graphical user interfaces. So in addition to introducing a central, well-organized repository to store all the imagery and texts needed to produce magazines and web site content, we had a secondary goal of reducing the number of people who had to learn to use the Book Club System through the terminal interface.

We moved all handling of products from the Book Club System to the new GUI and DAM-based system. Titles, prices, campaigns, sales items etc. are all handled through the Web Front End. Hence the editorial and marketing staffs do not have to learn the Book Club System or be dependent on system operators to get things done.

The new system outputs XML tagged text, which is easily imported into InDesign. This allows the designers to fully concentrate on design and layout, knowing that the quality control of data has been handled by the new DAM-based system.

**Making the integration**

It is easy to say, let's integrate. In real life there are a lot of considerations to be taken into account. The obvious are: people, time and money. But when you build an architectural structure like this, you want it to last. You want it to be flexible and robust enough to take you through a flow of new technologies, and an ever changing world of new business requirements.

In addition you also want to save money in future projects, by enabling new functionality to fit into the architecture. The basic idea is to make each component independent of the others. In that way one component can be replaced with no or minimal impact on the rest. Computer professionals have endorsed this approach for decades, but it's been easier said than done, until now.

Recent development and standardization has made the old dream come closer to reality. I'll describe our approach and experience.

## Choosing tools and methods

The most difficult task is often to choose the tools and methods to use. There are so many options and so many fanatic believers in the various camps, that making these decisions can cause anxiety. But a choice has to be made and more important, when a choice is made, stick to it! Our strategy is to keep the number of different platforms as low as possible, whether it's operating systems, hardware, development tools or methods. We want to focus and be very good in something - instead of knowing a little bit about everything.

Yet, be warned about the "hammer trap". When all you've got is a hammer, everything starts to look like a nail. So it's all in the balance between the one and the other extreme.

We use Oracle as the primary database platform, Windows servers and IIS with applications developed in Visual Studio 2003/2005. Web Services using SOAP (Simple Object Access Protocol) is the primary choice for integration between systems. Oracle is there because our legacy systems use it, and it was the obvious choice at the time. And we run it on 64-bit Linux. As always, there is a need to be pragmatic and realize that you're always in transition.

Our DAM system is Artesia[1] for DAM (previously known as Teams). The Financial System is from Schilling Data[2] (previously known as Oase) and the Logistical System is Microsoft Dynamics AX[3] (previously known as Navision Axapta).

That is where Web Services comes in handy. The database and operating system becomes less relevant, as long you access the system's functionality with Web Services. In fact you can even mix and match which services you run in-house and which you leave to others.

## Service Oriented Architecture

Far from all standard systems provide Web Services yet. So the ideal world, often described as Service Oriented Architecture (SOA), is still years away. But there is no reason to let the perfect be the enemy of 'good enough'. So we have applied various other methods while waiting for all our vendors to provide proper Web Service interfaces to their systems.

## Changing logistical system

We knew that we would have to change the Logistical System after we integrated the DAM system, so we wanted to make the system-to-system interfaces prepared for the change. To accomplish that we made Web Services interfaces into our old logistical system and isolated all code to make the transition as easy as possible. The new Logistical System will have Web Services in the next release. In the mean time we had to do some changes in order to integrate it, but as the methods were similar, it was a doable job.

## Integrating IBM mainframe

The Book Club System is run by an external company in Sweden on an IBM mainframe. The best way to get online access to the system was through Web Services. In 2000 using REST (Representational State Transfer) and returning XML was the most feasible way. The

---

[1] *Artesia*: develops solutions for digital asset management, their main product being *Artesia DAM* (formerly *Teams*). A company within The Open Text Digital Media Group.
http://artesia.com

[2] *Schilling Data*: a Danish company, which specializes in developing applications for the publishing industry.
http://www.schilling.dk

[3] *Microsoft Dynamics AX*: http://www.microsoft.com/dynamics/ax/default.mspx

interface is quite simple. The request is sent as HTTP GET requests and the results are returned in XML.

**Using what's there**
Our Digital Asset Management system is based on Artesia's solution. It comes with a J2EE-based API. Until Artesia delivers their announced Web Services interface, we use J-Integra[4] to bridge between IIS, .NET and Artesia. The APIs were historically on a rather low level, requiring many calls and surrounding logic. With Web Services it makes sense to create more high level functions.

**Making interfaces that not exists**
Besides using SOAP, we've made interfaces based on database tables. It works like this: the sending system inserts records into a table, and the receiving system updates its own database according to defined rules. By putting Web Services between the database tables and the sending system, we are able to provide flexibility for the future.

**Going one step further**
We have made a pilot system where all functions are exposed as Web Services. So the user interface can in reality be a standalone application using these functions. That makes it possible to have one user interface internally, while other internal systems, customers or partners can use the same functionality integrated into their applications.

**Conclusion**
How various systems are integrated and interconnected will vary from company to company. There are many legacy systems involved. Creating an optimized architecture depends on what's in place and what's missing. In an ideal world systems are well defined with regard to what they do and how the data is stored. And by tying them together with machine-to-machine interfaces like Web Services, one can use functionality from one system in another.

The purpose of the DAM system will vary. In our case we aimed at having a central storage for all our product assets stored in one place.

By separating the user interface from the underlying systems and using clear and well defined services, the DAM system has become an integrated part of the whole portfolio of systems. Not just a system for a small group of people, but a part of the toolset which supports many different business processes.

Assets from the DAM system are made available to other systems where they are needed.

Not only company internal systems can be integrated. Customer and vendor systems can be integrated as well, given that the overall architecture can support it. The options are unlimited.

[4] *J-Integra*: A software package developed by *Intrinsyc Software*, a company which specializes in bridging middleware for interoperability between *Java/Corba* and *Microsoft COM/.NET/Exchange* applications. http://j-integra.intrinsyc.com

## REFERENCES

### Literature:


- *Service-Oriented Architecture: Concepts, Technology, and Design*
  Thomas Erl, Prentice Hall (2006)
  ISBN: 0-13-185858-0
  http://vig.prenhall.com/catalog/academic/product/0,1144,0131858580,00.html

- *Understanding Enterprise SOA*
  Eric Pulier and Hugh Taylor, Manning Publications Co (2005)
  ISBN:1-932394-59-1
  http://www.manning.com/pulier

- *Enterprise SOA: Service-Oriented Architecture Best Practices*
  Dirk Krafziq, Karl Banke & Dirk Slama, Prentice Hall (2005)
  ISBN: 0-13-146575-9
  http://vig.prenhall.com/catalog/academic/product/0,1144,0131465759,00.html

- *Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services*
  Thomas Erl, Prentice Hall (2004)
  ISBN: 0-13-142898-5
  http://vig.prenhall.com/catalog/academic/product/0,1144,0131428985,00.html

- *Understanding SOA with Web Services*
  Eric Newcomer, Addison-Wesley (2005)
  ISBN: 0-321-18086-0
  http://www.awprofessional.com/bookstore/product.asp?isbn=0321180860&rl=1

- *Understanding Web Services: XML, WSDL, SOAP, and UDDI*
  Eric Newcomer, Addison-Wesley (2002)
  ISBN: 0-201-75081-3
  http://www.awprofessional.com/bookstore/product.asp?isbn=0201750813&rl=1


### Web sites:


- Intro to Digital Asset Management: Just what is a DAM?
  (→ Integrating DAM into your broader CM Framework)
  http://www.cmswatch.com/Feature/124-DAM-vs.-DM?printable=1

- Building Web Services the REST Way
  http://www.xfront.com/REST-Web-Services.html

- REST:Representational State Transfer(Wikipedia)
  http://en.wikipedia.org/wiki/REST

- What Is Service-Oriented Architecture?
  http://webservices.xml.com/lpt/a/1292
  http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa_p.html

- Service Oriented Architectures (Wikipedia)
  http://en.wikipedia.org/wiki/Service-oriented_architecture

- Web Services and Service Oriented Architectures
  http://www.service-architecture.com/

## Examples of Web Services Methods

This table shows the major Web Service methods we needed in order to integrate the Editorial System. The main job of course, is to define these services in a flexible and robust way, so they can serve as general purposes as possible. In reality one may need more methods than listed here, but this is only to show the general idea and basics.

**Example of DAM system services**

| | |
|---|---|
| PutAsset | Upload an Asset to the DAM system. |
| GetAsset | Get an Asset from the DAM system. |
| SetMetadata | Set the Metadata for a given Asset |
| GetMetadata | Get the Metadata for a given Asset |
| DeleteAssetandMetadata | Delete both the Asset and its Metadata |
| SearchAsset | Search for Assets at give criteria and return a list |
| CheckOut | Mark an Asset as Checked out |
| CheckIn | Mark an Asset as Checked in |
| Export | Export an Asset to a given location |
| ExportAndTransform | Export an Asset and do a transformation |
| CreateCollection | Make a collection of Assets in the DAM system |
| AddToCollection | Add a given Asset to a collection |
| GetCollection | Retrieve Assets in a collection and their Metadata |
| RemoveFromCollection | Remove an Asset from a collection |
| ExportCollection | Export Assets and/or Metadata to a given location |
| DeleteCollection | Delete a collection (not the Assets) |

**Examples of services in the Logistics System**

| | |
|---|---|
| NewProduct | Create a new product in the system with given data |
| UpdateProduct | Update data about a given product |
| GetProduct | Get product data |
| SearchProduct | Search for products on given criteria and return list |
| GetProductSales | Return defined sales figures for the product |

**Example of services in the Financial System**

| | |
|---|---|
| NewProject | Create a new project in the system with given data |
| UpdateProject | Update data about a given project |
| GetProject | Get project data |
| SearchProject | Search for projects on given criteria and return list |
| GetProjectCosts | Return defined cost figures for the project |

**Example of services in the Web Publishing System**

| | |
|---|---|
| NewProduct | Create a new product in the system with given data |
| GetProduct | Get product data in the system |
| UpdateProduct | Update the product data |
| DeleteProduct | Delete a product from the system |
| NewAuthor | Create a new author |
| GetAuthor | Get data about an author |
| UpdateAuthor | Update data about an author |
| DeleteAuthor | Delete an author from the system |
| NewText | Create a new text item with additional data |
| GetText | Get a text and corresponding metadata |
| UpdateText | Update a text and its data |
| DeleteText | Delete a text item and its data |
| Publish | Set publishing status for a product, author or text |
| ConnectItems | Make connections between products, authors and texts |