

Architectures for System integration

Chris Chambers

BBC R&D and EBU Project Group P/MDP

Broadcast production systems typically were built as a collection of “silo” or “island” devices. However, with the increasing use of IT in production, and the sharing of common resources such as storage and connectivity, the question becomes: *how best can we glue all this equipment together into an integrated system?* This is exactly what EBU Project Group P/MDP has been studying.

It is a fact of life that there will rarely, if ever, be global agreement on common standards in some fields. Which nation will give up their national language for a universal “Esperanto”? Technology is no different. Broadcasters will remember the “camps” of PAL and NTSC.

Accepting this fact leads us to look for solutions that allow interchange between different “standards” – what we choose to call *Middleware*. In the case of PAL/NTSC, the answer was the standards converter – an early example of middleware.



Figure 1

A familiar problem with a well-known solution

At a more basic level, consider the variety of different mains

power connectors which exist throughout the world. To enable your equipment to work anywhere, it is necessary to have a different mains cable for each type of connector. And then came the middle-ware – the universal mains adapter.

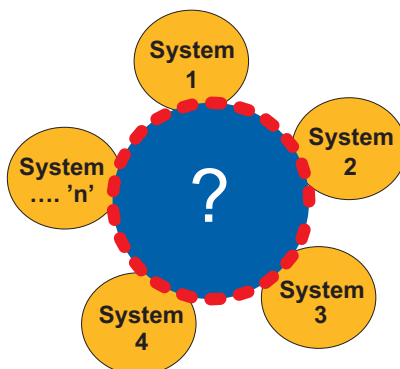


Figure 2

Where the universal adapter is needed

In broadcasting these days, every system tends to be a framework; with endless possibilities, but only working if tightly integrated with all of the other systems (frameworks) of the organization. This can be very hard to achieve.

Even if you do reach some level of integration, full interoperability is seldom achieved. It tends to be expensive, takes a lot of time to configure and implement, and fails to fully realize the opportunities for flexibility. On top of that, the demands to constantly change and optimize our workflows using these systems are increasing.

We are missing the universal adapter between our systems!

Scope of this article

For the purpose of this article, middleware should be seen as a collection of technologies which combine applications, or parts of applications, in a common and structured way. Middleware doesn't achieve this alone. The management of middleware and the actual integration are of equal importance. For that reason, the P/MDP¹ Group set the scope of its work to be: **System Integration Architectures**

Middleware is but one (vital) element in this environment. A collection of tools is the glue.

Scope of use: the value chain

Most broadcasters have a two-tier value chain consisting of Services and Programmes, illustrated in Fig. 3. System integration architectures and the different middleware technologies affect the processes in this chain as well as the supporting processes.

The integration architecture ensures that information and functions are accessible from all the process stages – from early season planning to play-out – via production, distribution and transmission. It is even relevant in the delivery platform where some data reaches the set-top box (e.g. the EPG).

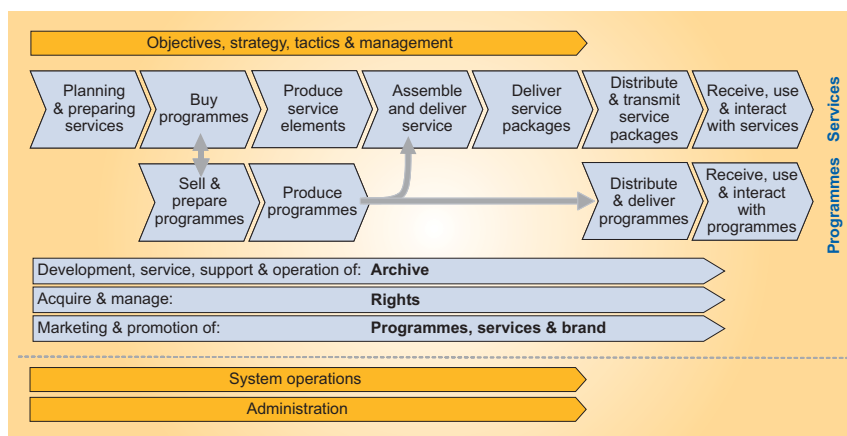


Figure 3
Typical broadcaster's value chain (source: DR)

Broadcast-specific scope

In many parts of our operations, we can benefit from integration techniques commonly used in other industries. But some areas are more critical to broadcasters than to other industries. This is because broadcasting deals with real-time rich media and not just common data in the form of text and numbers:

1) Media asset management

Storage and archiving of our products: video, audio and interactive services. This is about the general integration architecture between our assets and all of our other systems.

2) The production environment

Where can benefits and new ways of working be realized using integration technologies?

3) The play-out environment

We need to ensure data and content integrity from planning and scheduling, the production, through play-out, to the end user.

4) Workflow management

We need to connect the workflows which relate to the processes in the whole value chain across systems or modules of functionality.

1. P/MDP stands for "Middleware in Distributed programme Production". The Group was created in autumn 2003 and consisted of various EBU members. The main contributions to its report and this article came from: the BBC, BR, DR, the IRT, NRK, RAI, RTR, SVT and VRT.

What is so great about a system integration architecture?

The arguments for your organization to establish and maintain a system integration architecture can be summarized as follows (this is not a prioritised list):

- **Flexibility:** fast changing work-processes and workflows need flexible access to functions and data. High barriers between systems are falling or changing to small boundaries as modularity is growing. A good system integration architecture and the appropriate middleware are helping to provide this flexibility.
- **Speed:** where functions and data have to be used across systems, middleware speeds development and changes (in the long term, but not necessarily in the short term).
- **Cost:** the costs of building and maintaining unique one-to-one integration between systems is rapidly growing for larger systems. Changes are expensive as they need to be applied to more than just the primary system. A hub-and-spoke approach is more efficient. Middleware can act as a hub.
- **Standardization:** as methods and interfaces are re-used, fewer components and skills are needed. Having a way of handling integration makes it easier to decide what you don't need to do.
- **Data integrity, reliability and robustness:** as more and more data are used across systems and different work processes, a system integration architecture and some middleware solutions ensure the data integrity across systems and situations of use. For horizontal systems, efforts spent in making the evolution reliable, secure and robust can be concentrated in the service layers rather than distributed over single applications.
- **New products and services:** easy and secure access to information across systems opens up opportunities for new products, especially on new delivery platforms.
- **Prevent lock-in by vendors:** you can replace a sub-system from vendor A by an equivalent system from vendor B, without having to reconfigure the rest of the system.
- **Overview:** as common ways of achieving integration are established, the organization gets a better overview; both due to the standardization and to the needed repository of information about systems, functions, data, integrations, interfaces and metadata. This opens up opportunities for stronger change management routines. More specifically, the ability to search within data across systems is improved.
- **System planning:** standardization allows technical planning departments to use tools which will make the planning process much easier. The focus will no longer be fixed on specific technical problems but much more on workflows, or rather on optimizing the workflows. Planning of complex broadcast-systems in future could be much more like "Lego™".
- **Training:** an understanding of the workflows in which the operators are involved will become more necessary in the future than today. Therefore, not only will dedicated technical training for a device or software application be necessary, but also education on understanding the whole production chain. This can become much simpler if the system architecture is based on generic and pre-defined processes rather than on proprietary structures.

Examples

Several EBU members have already gained experience with system integration architectures:

- **Bayerischer Rundfunk** is implementing an IT-based play-out and production centre. It encompasses a messaging-based system for system integration.
- **Danish Radio** is in the process of moving from a vertically to a horizontally integrated system architecture. Standards and policies are established and both a broker and a concept for service-oriented integration are in operation. This has been done in close connection with the implementation of a DR metadata specification.

- **BBC Research & Development** is studying a completely horizontally integrated production architecture. This project can be regarded as having the luxury of a “green field” situation, trying to utilise the benefits of horizontal system integration to its fullest extent.

Key concepts

We are used to working with isolated systems, interfaced by *people*. These are systems in *co-existence*. Today this is changing. The main task of system integration today is to turn co-existing systems into *co-operating* systems, to collect systems into “super-systems”, when appropriate and efficient, and to optimize workflows.

Introducing layers

In the late nineties the primary view was that one had to connect all systems or modules to each other directly. However, for a large number of modules, this becomes almost unmanageable. The alternative is to connect the modules via some common glue. But this glue needs to be well-defined and widely accepted. The development of a layered system architecture was a good first step.

A system can be seen as split into layers in a *horizontal architecture* where each layer is independently accessible. Systems without this approach are referred to as *vertical systems*.

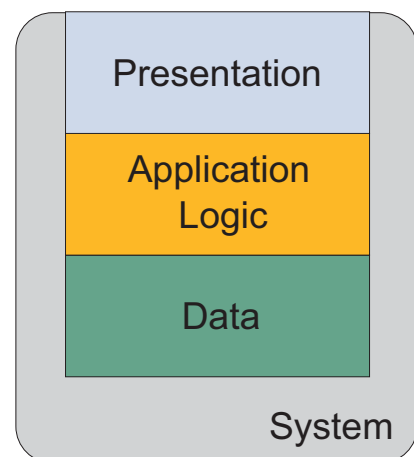


Figure 4
A layered system

How many layers?

In this horizontal approach, the number of layers can differ. Therefore it is generally called an *n-tier horizontal architecture*, where n indicates the number of layers.

Usually the design starts with three layers, each with its own tasks:

- 1) Presentation layer -> provides input & output for the system;
- 2) Application Logic layer -> the “brains” of the system;
- 3) Data -> the memory or content of the system (databases and file servers).

In many situations the concept can benefit from introducing a layer under the databases and file servers, taking into account that physical storage can differ. This then allows you to change or scale the storage without changing the data layer. So the fourth layer would be:

- 4) Storage -> the physical place where data are put.

Open or closed systems

It is important to realize that there might be no connection between the actual construction of a system and how it appears or reacts to the outside world. A system using the latest “horizontal” technology² can still appear as a closed vertical system. The constraints may be imposed by unpublished or licensed interfaces, or by proprietary additions to known application interfaces (APIs), protocols, etc.

2. Such as portal servers, web-based user interfaces, modularised business logic on application servers, and XML data servers.

On the other hand, some older and non-layered systems have quite open interfaces, although there is a tendency to refer to non-layered systems as vertical and closed.

In the real world, we are dealing with a complex mixture of scenarios, and the solutions tend to be equally diverse.

Component-based systems

The layers mentioned are an abstraction, introduced to illustrate logical groups within systems, which in development and daily work are useful to treat separately.

We can go one step further and describe systems which are component-based³.

“ A component is a “black box” with known and described interfaces which can be used without knowing anything about the internal structure or internal functionality of the box. ”

These components can be grouped inside the above-mentioned layers, although some components work across layers as well. As long as they stick to the rules in the above definition, that should not be a problem.

Some of the components' interfaces should be openly accessible, but not necessarily all. This can be achieved in many ways and is one area where the need for standardization arises. The main point to remember is that **systems should be component-based**.

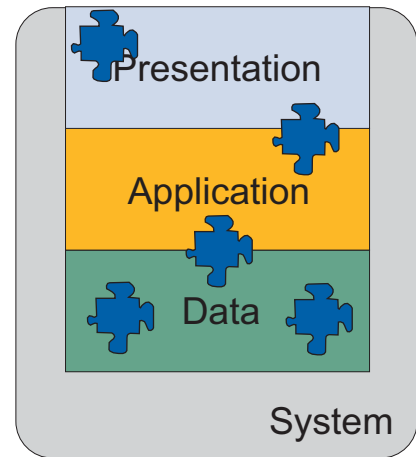


Figure 5
A component-based system

Middleware: the glue

To solve the problems mentioned at the start of this article, something was needed to combine all these layers and components in a flexible and efficient way – some kind of “glue”. This was originally referred to as **middleware**, already suggesting that no one really knew clearly what it was.

One of the early techniques to act as glue was the data-broker (a hub and some adapters to the systems). Without going into technical details, the concept and expected benefits are illustrated in *Fig. 6*.

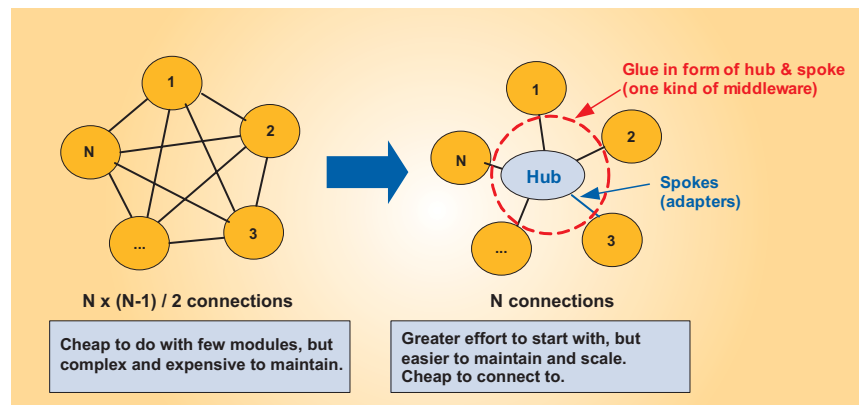


Figure 6
Example of glue

Fig. 7 combines the concept of middleware (glue) with the layers introduced earlier.

The way the middleware interconnects the data layer with the application (logic) and the presentation to the user, is by using *communication via interfaces*. These interfaces form the boundaries of

3. The term *component-based* is preferred to *object-oriented* because object-oriented is just one example of a component-based design, e.g. a procedural design can also be component-based.

the “puzzle” pieces in *Fig. 7*. It is important that the interfaces allow for communication in a common, flexible, structured, efficient and re-useable way.

Where do components fit into this?

As we discussed before, systems are nowadays more and more component-based. *Fig. 8* extends our layered model with this notion.

The components are drawn as little puzzle pieces, indicating they have their own interfaces. This means we can speak about interfaces at multiple levels: for example, the interface to the data-layer or the interface to a component within that layer. Components could be system components or middleware components. Working at component level is leading us to another kind of glue – indeed another integration technique: Web services.

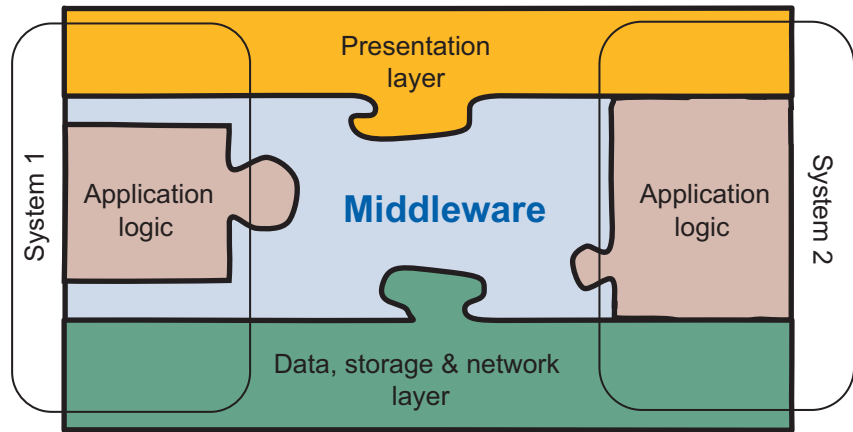
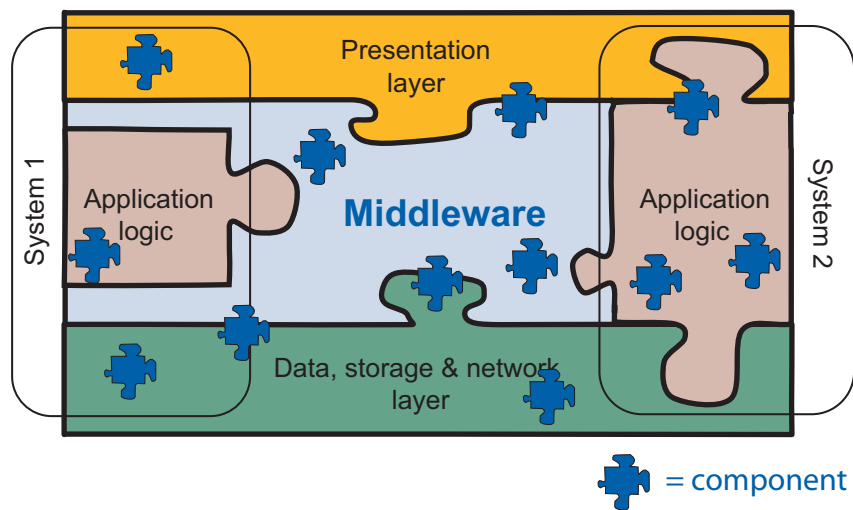


Figure 7
System concept, including interfaces
(based on a diagram by Bob Edge, Thomson GVG)



 = component

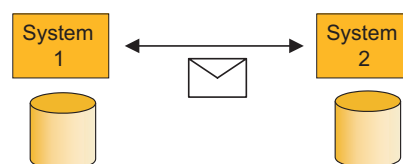
Figure 8
Systems can be built out of many small components

Most common integration types

The resulting system integration architecture depends on your particular business model (the products and operations) and your particular IT environment. Here we describe four common integration types.

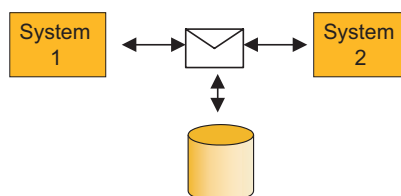
- **Message-based integration** (copying data between systems)

The same data is needed in different systems and an automated exchange mechanism provides copies of data in these systems and ensures data-integrity while doing so. It can be set up as rule-based or event-based. Functionality is made available by an integration broker.



- **Data-carrying integration** (Different systems access the same data)

Data exists as a single copy in a single location. All systems work real-time on the same instance of data. Integration is achieved by agreement between the systems about the data model they use, and about a common logic for reading and updating data. This is provided by a data-carrying integration platform, primarily consisting of an application server and a database.



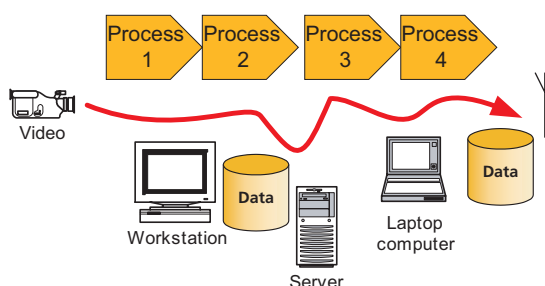
○ **Portal integration** (Wrapping of multiple user interfaces into a single one)

Employed where users require a more integrated user dialogue or interface than is provided by the separate applications. Where individual customizing of the user interface is required, or where certain functions should be automated, or where single logon is needed ... an integration of the user interface is necessary. Web-based user interfaces provide integration via a portal.



○ **Workflow** (Mechanisms to co-ordinate events in systems)

Refers to automating or semi-automating of workflows, by integrating functionality and data. To support tightly connected or integrated business processes it should be possible in advance to define which actions need to take place when certain data are exchanged or updated. Workflow mechanisms can be found in each of the three former described integration types.



The above integration architecture concepts are related to planning, production, and play-out systems, as well as the administrative back-office, financial and accounting systems. When it comes to executing play-out or transfer of real-time, rich multimedia files (e.g. a 50 Mbit/s video file) other tools and technologies than described above are used. However, the management, control and reporting of this can be handled like any other kind of data in any other industry.

Middleware services for broadcasters

When setting up broadcasting facilities, this often means we have to solve recurrent problems. We have learned that concepts and technologies are available which enable us to reuse solutions for those problems. An example of such a concept is the **Services-Oriented Architecture**.

When using a Services-Oriented Architecture, one may ask the question whether there are special middleware services needed for broadcasting.

Let us first try to find out what **middleware services for broadcasters** actually are.

Abbreviations

AAF	Advanced Authoring Format	PAL	Phase Alternation Line
API	Application Programming Interface	SMPTE	Society of Motion Picture and Television Engineers
MXF	Material eXchange Format	UML	Unified Modelling Language
NTSC	National Television System Committee (USA)	VTR	Video Tape Recorder

Understanding services

A service is basically a special system component which can be invoked by other components wanting a certain task to be performed on certain data, e.g. an authentication service which is granting or denying access to resources. To be able to do this, it is necessary for the invoking components to know what operations are available and how to ask for these operations. The specification of these is called a *service interface specification*.

How should such a service interface be specified?

The most efficient and reliable technique is to use modelling methods; tools and technologies supporting the task of defining the boundaries and behaviours of software components.

A useful strategy in designing a service architecture is to distinguish services into those specific to your industry, the *broadcast domain*, and those which are not specific to a particular industry, also known as *pervasive services*.

Broadcast domain model

A domain model describes the functional elements used in order to provide solutions *specific* to a *business domain*, in our case the broadcast domain. It should not describe the internal workings of the (functional) elements, nor provide much detail about them. Instead it must just encapsulate the essential aspects related to the domain. The functional description should be limited to that of the interfaces and should include the static definition as well as the dynamic behaviour.

A broadcast domain model should define all “bread & butter” services specific to the broadcast domain, e.g.:

- Essence transcoding;
- Speech-to-text transcription;
- Material copies management;
- Material stream control.

The task for the users within a certain business domain is to specify their domain model and to benefit from existing standards. This may be done in collaboration with vendors. The users' involvement is particularly important since the domain model reflects the business which the users know best.

Pervasive services

Pervasive services provide functionality which is *not specific* to the broadcasting needs. Pervasive services are reusable by other services and applications. The use of pervasive services allows us to benefit from solutions worked out in other industries, for example:

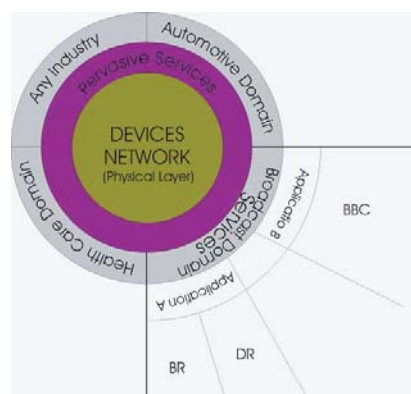


Figure 9
This figure shows different industry domains, including a (currently hypothetical) Broadcast Domain. The “BR”, “DR” and “BBC” slices illustrate different broadcasters' usage of it.

- Authentication;
- Session management;
- Time references;
- Workflow management.

For the pervasive services, the users should select available services and implementing technologies which are appropriate to support their business. The proper requirements have to be applied to the services (e.g. the bandwidth needed and the desired network characteristics), to guarantee they meet the users' specific operating conditions.

Different views

There are different types of players involved in the development of IT-based TV programme production systems. These different roles require different views of the system which is modelled, for example:

- **Planning Engineer View** – *a set of service definitions and high-level descriptions of their interactions.*
- **System Integrator View** – *a description of how to use a service.*
The System Integrator View has to represent a high-level view of the interfaces of the services and their dependencies on other services (e.g. the pervasive services).
- **System Developer View** – *all the detailed information about the components to be implemented.*

Modelling in the broadcast environment

Modelling is a widely used method in IT system design. As the broadcasting industry is making more and more use of software solutions, the use of modelling techniques becomes more important as well.

The use of models to describe technical systems is not a new idea. Block diagrams and connection schemes are models also. The difference is that, with IT modelling techniques, the diagrams represent software systems and not necessarily physical objects.

Unified Modelling Language

UML provides different types of diagrams for the static design and the dynamic behaviour of IT systems. A brief introduction to the use of different UML diagrams is provided below, using an example which describes the process of recording with a VTR. Arguably the best known modelling methodology is called UML (Unified Modelling Language). In the last 10 years it has established itself as the predominant formal notation for the description of IT projects.

Use Case diagram

A Use Case Diagram gives an overview of what happens in a 'business process'. It does not explain how the result is achieved, but only what happens, regardless of any specific product or solution. Use Case diagrams are used to describe the interaction of actors and systems in two ways:

- 1) A drawing with formal icons for actors (representing roles, humans or systems, but never a job description) and activities.

- 2) A formalised text part defining the actors, the preconditions and the business rules relevant for the Use Case.

Several Use Cases can be collected into logical groups to form a drawing of a Use Case scenario, including text parts for each Use Case.

There are frameworks which give some guidance in how to deal with system analysis and description. For more information look for MDA in [1] and Zachmann in [2].

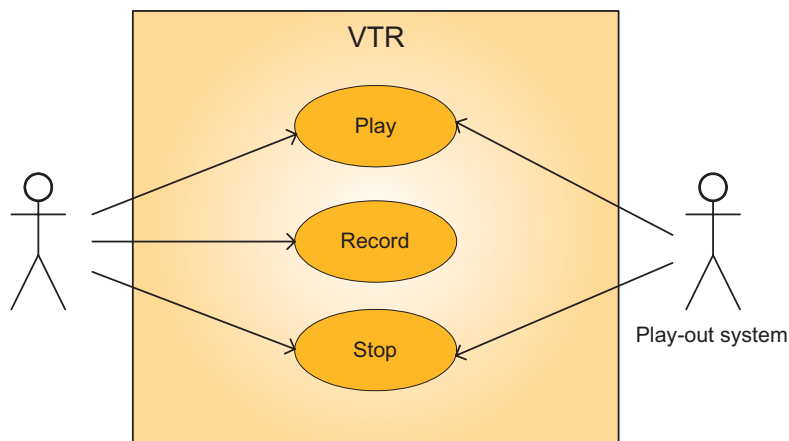


Figure 10
Use Case scenario for use of a video system

The vendors' perspective

Types of vendors

Current broadcast-system vendors form a heterogeneous group. Basically, there is a group of traditional broadcast vendors and a group of IT-originated companies, each approaching the customer from a different angle.

Some traditional broadcast parties seem to have a stronger tendency to think in silo-based vertical systems, while the other extreme is formed by some IT companies entering the market virtually without knowledge of broadcasting, but equipped with a bag overflowing of IT standards nomenclature. The first can be argued as undesirable due to its inherit “lock-in” character, while the second may not prove any better as there still could be a “data lock-in”. That is: the system may be open, but if you don't understand the data model inside, you still cannot use it.

The foregoing sketches out the extremes but, in practice, both “camps” are moving towards a better understanding of the customers' requirements (as is the customer himself). It seems that some smaller companies in particular have a good nose for understanding the customers' specific requirements.

Key observations

Vendors are opening up their systems

Openness is the norm in IT, but is only slowly becoming practice in the broadcast domain. One reason for this is the large influence of traditional wholesale providers. The main driver to open up is, quite simply, pressure from customers demanding openness.

Vendors put themselves in the centre

When vendors claim to interoperate with others, they often put themselves in the centre of the system. Instead of offering a single service to the outside world (like plug-ins for your browser), they argue that “others should interface with us”.

Middleware is known, but interoperability limited

Vendors are using middleware, but often only in their own product suites. For commercial reasons it is not exposed to the outside world. Another reason is that there is a lack of semantics (“What does the data this service provides mean?”). The net result is that interoperability between broadcast applications using middleware is limited.

Vendors are reactive

Vendors are reactive and not proactive; they will only implement when there is a demand (customer project driven). The broadcasting market is relatively small and each broadcaster poses different customisation requirements and/or a different expression of the same requirements. Some vendors also remark that the broadcast industry is special in that much R&D is done by the customers themselves, which is not the case in the IT industry.

Middleware reliability is ok, but at a cost

In general the vendors do not see difficulties with middleware reliability, but they do warn it may not be the best solution for each problem. The engineering effort to provide high reliability may be higher than for traditional systems. Also, reliability should not be the single reason to use middleware (but, for example, scalability + reliability).

Managing system integration

The cost for system integration can differ dramatically depending on how the work is managed. At one end of the spectrum every integration is a stand alone solution with only one person capable of handling development, maintenance, etc. At the other end, a corporate integration architecture is implemented.

To boost economy, quality, capability, etc there is a need for standardization and streamlining of the integration efforts. It is important to have a strategic scope that spans more than the current projects. If possible this responsibility and governance should be centralized within the company – the consultant company Gartner has named such a function Integration Competency Center, ICC.

As already stated, this is very much about managing and control. To deal with all this increasing complexity and variety is much like working with quality assurance programs:

“ Describe what you do and do what you describe. ”

To move away from the situation where specialists are formed around each isolated system, and into a world with a large variety of skills across systems, components and middleware, is a huge effort.

Standardization

There are two areas where standardization efforts would be welcome:

1) **A common (core) metadata model**

There is a lack of a common (core) metadata model. This is the *top priority* issue for most vendors. Existing metadata specifications have not been very successful and are seen as too large/vague to apply. There is a strong demand for a more modular and business-oriented approach: e.g. a simple subset with a coherent structure. This must include the semantics (= unambiguous meaning) of the elements it provides.

2) **“Bread & butter” services**

The business services commonly used by broadcasters, such as ingest, play-out and scheduling are the broadcasters' “bread & butter” services. It seems that standardization of middleware / system integration architectures could best take place at this level. These primary services could all be broken down into non-competitive and common technical services (play, record, transfer, transcode, etc.) and all have a need to interface. Besides services, relevant business objects should be specified as well. The specifications should not dictate technology,

but allow for competition. Once the services are defined, it should be investigated if toolkits, APIs, etc., would be of value.

Who should standardize this?

A combination of an IT organization (e.g. OMG [1]), together with a broadcast organization (e.g. the EBU or SMPTE) would be preferable. Some manufacturers noted that they would like to see a community effort using more Internet-based tools and less physical meetings, which would also allow smaller parties to be involved: e.g. an open forum on interoperability.

When should it be standardized?

It seems there currently is not a strong urgency for standardizing middleware-related topics. Middleware has been around for some time already (some claim for 15 years already) and as one manufacturer put it: **“there is no immediate penalty if you don't use it”**.

Also many manufacturers have just started to implement other important technologies, such as AAF/MXF support in their products, and they need time to allow enough Return On Investment to be generated.

However, in view of the developing convergence of traditional broadcasting techniques with IT ways of working, and also because any development today has a strong business element, users now require the ability to specify IT-based applications and business structures within their organizations. It is now a matter of urgency that standardization work begins, to enable users to specify functionality between products.

Golden Rules

The experiences shared by the P/MDP Members resulted in a list of “Golden Rules” for broadcasters, vendors and standards organizations. Ten are listed below – but see the full report [3] for the complete list.

Table 1
Ten “golden rules” for system integration

#	Golden Rule	Notes
1	Middleware is infrastructure	Treat it as such: e.g. financially, management
2	Don't talk systems, talk services or functions	Otherwise you will see a misleading picture
3	Clearly define your process workflows	Look at middleware from the business perspective
4	Know your world	Set up a central information repository
5	Own your world	Define your vital business (objects) yourself
6	Partner with your vendor(s)	Make sure it is clear where the risk is borne
7	Demand open modularity	Proprietary combinations will lock you in
8	Demand open standards	Proprietary interfaces are too limited
9	Demand free choice of storage	Not one which is tied in with other components
10	Take real-time seriously	Plan for it from the beginning, don't use quick fixes



Chris Chambers joined the BBC as an Engineering trainee and this year is his 37th with the Corporation! Over the years, he has held a number of posts covering the installation of broadcast facilities in Outside Broadcasts, Studios, Broadcast Infrastructure and Data Services. He has also been involved with the strategic development of broadcasting systems, looking into ATM- and IP-based networking developments as well as file standards for broadcasting use. He has been a participant on a number of internal BBC groups covering these topics and has produced a number of papers on these subjects.

Currently in his post at BBC R&D, Mr Chambers runs a small team investigating networks and storage. He has been heavily involved in developments using ATM network structures to support live broadcasting within production areas, along with working on new standards within the AES and the IEC to support the audio on such systems. A part of the work of his team is to support the Desktop Production project within BBC R&D that aims to develop systems for complete broadcast and production structures using standard IT components.

Chris Chambers continues to represent the BBC in EBU project groups and currently chairs two of these, one working on the development of middleware technologies (P/MDP) and the other covering common control strategies for live structures (N/CNCS) – both of these areas can be applied to production and broadcast systems. He also chairs an AES standard working group on developing metadata structures and is currently working on two IEC standardization projects.

References

[1] <http://www.omg.org>

[2] <http://www.zifa.com>

[3] EBU doc. Tech 3300: **The Middleware Report**
EBU doc. Tech 3300-s: **Supplement to the Middleware Report**
Available via http://www.ebu.ch/en/technical/publications/tech3000_series/
