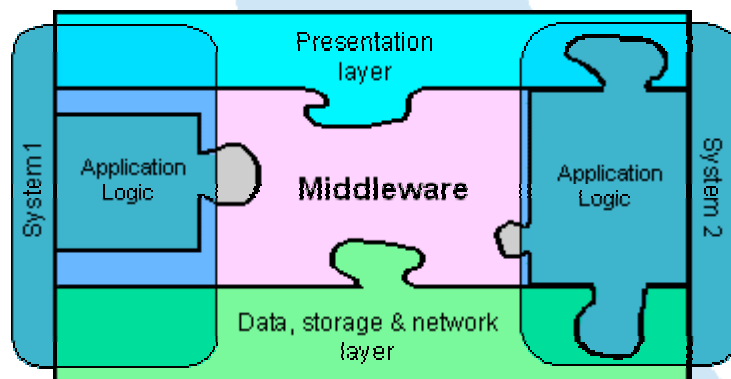


The Middleware Report

System integration in broadcast environments



Foreword

This document is the final report of the EBU project group on Middleware in Distributed programme Production (P/MDP).

It is intended to be of use both to IT people working in broadcasting and to broadcasting people working with IT, and to both managerial and technical in either field.

Some parts of it will seem over-simplified and some parts over-complex, but they will be different parts for different people. This is the inevitable consequence of trying to produce such a broadly accessible document.

Further, it will be seen differently depending on how far along the path of 'system integration' that the reader's organisation has progressed. It might be a welcome source of information to supplement an existing development or it might be an alarming introduction to what the future holds.

The EBU P/MDP Project Group was established in the autumn of 2003 to study the topic of Middleware, as the term 'Middleware' is used extensively in technical discussions but often lacks a clear definition.

Triggered by this, the group started out to define Middleware terminology as well as to produce an in-depth study of the problems and the opportunities. In doing so, it became clear that what the group was really studying was 'System Integration Architectures'.

The group subsequently gathered Members' experiences and held meetings with manufacturers' representatives to discuss this topic¹. Requirements for Middleware were identified and recommendations for future work were defined, as were important elements of working with System Integration and also some 'Golden Rules'.

This document is the result of their work.

Reading this document

The document is split into two parts.

1. **Tech 3300**
the main report is intended for a general readership. At the very least, read the Executive Summary.
2. Tech 3300s (supplement)
this part is intended for specialists, R & D departments, and managers with special interests.

Both parts contain glossaries; as terms and definitions are very important in this work.

Please note that the chapter numbering in the supplement follows on from that in the main report.

Level of detail

The reader should be aware of the way that this work uses illustrations and metaphors. It is working at the edge of what everyday language can describe. Therefore, in trying to communicate the implications of some very complex structures and information as simply as possible, it has to use some terms which, to a specialist, might be seen as imprecise or inaccurate.

The reason for this is to try and make accessible some very important concepts - to describe that of which broadcasters should be aware, and what they should do, when working with System Integration Architectures.

Some of the more detailed definitions and descriptions which have still to be defined, are suggested as future work.

¹ The P/MDP Group wishes to thank all participating vendors for their valuable contributions.

Table of contents

Foreword	3
Reading this document	3
Level of detail	3
Executive summary	7
Why is this relevant for broadcasters?	7
What is meant by Middleware?	7
Is there only one Middleware?	7
Is there a standard Middleware?	7
What do manufacturers think of standardisation?	7
What do manufacturers offer now?	8
What do they promise for tomorrow?	8
Typical Q & A	8
Recommendations	9
Overview and introduction	11
1. Cheaper, faster, and more flexible solutions!	13
1.1 Background.....	13
1.2 Key concepts	13
1.3 Integration: How to 'Glue'?	16
1.4 Scope of this work.....	18
1.5 What is so great about a System Integration Architecture?	19
1.6 Examples	20
1.7 Most common integration types	21
1.8 Middleware services for broadcasters.....	22
1.9 Modelling in the broadcast environment	24
1.10 Experiences from the broadcast world	26
1.10.1 Project A Bayerische Rundfunk (BR).....	26
1.10.2 Project B Danish Radio (DR)	27
1.10.3 Project C BBC's DP (Desktop Production) project.....	28
1.11 The vendors' perspective	29
1.12 Managing System Integration.....	30
1.13 Standardisation	30
1.14 Requirements	31
1.15 Golden Rules.....	31
1.16 Future opportunities	31
1.16.1 The transport layer.....	31
1.16.2 Grid computing.....	31
1.16.3 Meta-Metadata.....	32
1.17 Glossary	32

Executive summary

The EBU P/MDP Project Group has made a study of what is loosely known as 'middleware' and how it concerns broadcasters, manufacturers and the EBU. These three pages are a brief summary of their full report.

Note: the group decided that the relevance of middleware in the broadcasting environment was in the context of System Integration and so this is the main focus of this paper.

Why is this relevant for broadcasters?

We can no longer justify nor afford the luxury of self-contained, specially developed, complex production systems. Commercial pressures dictate that we must use and choose from a variety of existing equipment and solutions.

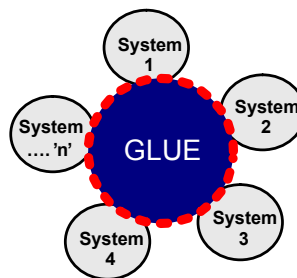
Also, to achieve the required flexibility across systems or parts of them, we are integrating more and more of our systems into new collections of functionality.

To avoid the restrictions and expense of proprietary 'islands' there needs to be a common, preferably standardised, way of interfacing equipment from different manufacturers.

This is where 'middleware' comes in. By providing well-defined, standardised layers between systems or parts of them, broadcasters are more likely to be able to share expensive structures, such as mass storage, amongst many applications.

What is meant by Middleware?

Middleware can be considered as software which 'glues' other systems together. It enables us to interconnect different application logic to networks and data storage.



Middleware also enables applications to interoperate with each other, enabling them to form and present different views of, and ways of accessing, functionality and data from and across each others' systems, regardless of their native platform or operating system.

Is there only one Middleware?

No. Actually the term is loose, as middleware can be built using different technologies and has different functionality. Middleware is a concept, like 'Interface'.

Is there a standard Middleware?

Yes, and no (as usual). Recognised standards exist but so do manufacturers' 'own-brands'. It is important to avoid becoming tied in to one manufacturer's proprietary implementation. Non-standardised, but well-specified, middleware can still be considered if it allows other manufacturers to connect to your system. Of course, the more standardised (and open) it is, the easier and cheaper current and future integration will be and the more choice users will have.

What do manufacturers think of standardisation?

We see the same two arguments as with other standards: manufacturers would like a proprietary system if they think they can lock-in customers but would want an open standard if they see a risk of them being excluded from the market if someone else's proprietary system becomes widely adopted.

What do manufacturers offer now?

There still seems to be a tendency for manufacturers to offer proprietary structures.

What do they promise for tomorrow?

Many promise products and an environment where users can plug new applications into a currently installed system and that these will inter-work seamlessly with the structure.

However, for this to be true across the wide range of manufacturers, the development and adoption of open standards is required.

Typical Q & A

How do you manage this more complex and differentiated landscape?

By using increasingly structured and reusable ways of making your integrations. Like quality assurance programs you should 'describe what you do and do what you describe'. And this goes for your business processes as well as for your systems and how they interact.

Be aware that this could be as much about change management in your organisation as about technology.

How do I go about integrating my systems?

1. Agree within your organisation what you are trying to achieve. Identify your important cross-systems processes and then the services you require from any middleware. This should enable you to:
2. Draw up an idea of the required system integration architecture, which is essentially a blueprint of how you wish all of your kit to connect together. Then:
3. Look at the equipment that is available, affordable, and capable of the necessary inter-communication and then you should be able to:
4. Choose a suitable implementation.

How do I go about determining my 'System Integration Architecture'?

The accepted technique for representing processes, functionality and the required data structure is to use modelling. For an illustrated example, see the full P/MDP report which includes one showing, with a step-by-step analysis, how to derive the middleware needs of an ingest application.

The full report also provides three real cases of how actual broadcasters solved their particular requirements.

Note: the 'modelling' approach might seem unfamiliar if you are not used to IT-modelling. See the Recommendations below.

Could you give me some golden rules?

Yes. For example:

Golden Rule	Key message
Clearly define the vital process workflows in your organisation	Look at middleware from the business perspective
Define your world as a broadcaster	Or you will be left alone with other parties definitions of your world
See middleware as a vital part of your infrastructure	Middleware IS infrastructure
Begin working with the supporting processes and then carry on with the main processes	System integration where well-known and common techniques are existing

(There are many more Golden Rules in the supplement, chapter 9.)

Recommendations

- Everyone - especially managers - should read the Introduction and Overview on the next few pages. Many will benefit from reading the full middleware report in the supplement and discussing it with colleagues. This Introduction and Overview is of approximately 28 pages and carries the most important messages. The supplement gives detailed information for the specialist and for people with special interests.
- Make sure that business units understand their own needs and can describe them as well as the technical departments can describe and implement the solution.
- Study process and system modelling, because it is almost an essential requirement before being able to understand how to design new systems. It would help to look at some of the many websites about the subject. For references see annex H (supplement).
- Learn about new technology, such as web-services, which promise better integration options and re-usage of software systems in your organisation.
- This document and the supplement is available for download from the following page:
http://www.ebu.ch/en/technical/publications/tech3000_series/index.php

Overview and introduction

Appropriate for both expert & non-expert readers

This section gives a high level introduction to the main terms and concepts of middleware. However it is important to note that many explanations given here are only one representation of several possibilities.

This reduced complexity is intended to aid accessibility to this complex subject. More complete, detailed and formal descriptions are given in the supplement to this report.

1. Cheaper, faster, and more flexible solutions!

1.1 Background

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 to 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.

At a more basic level, consider the variety of different mains power connectors that exist throughout the world. To enable your equipment to work anywhere, it was necessary to have a different mains cable for each type of connector. And then came the middleware - the universal mains adapter.

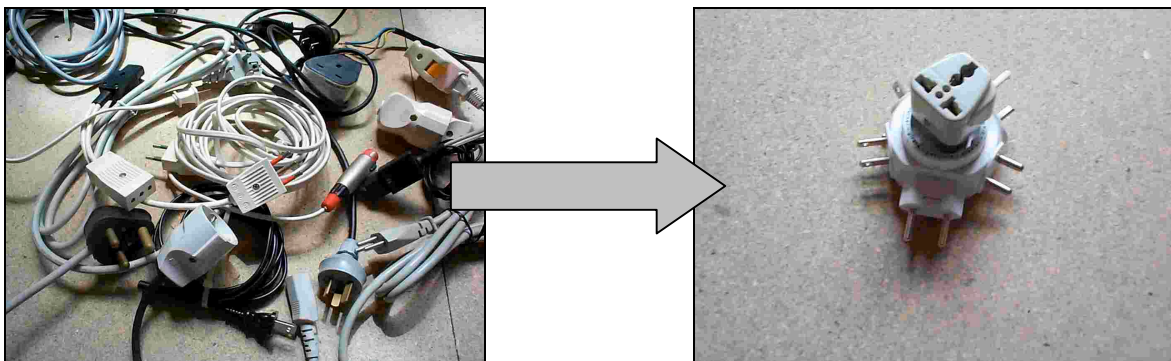


Figure 1.1 A familiar problem with a well known solution

Back in broadcasting, these days, every system tends to be a framework; with endless possibilities, but only working effectively if tightly integrated with all of the other systems (frameworks) within your company. 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 realise the opportunities for flexibility. On top of that, the demands to constantly change and optimise our workflows using these systems are increasing.

We are missing the universal adapter between our systems!

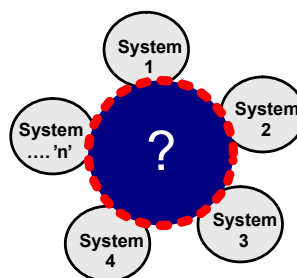


Figure 1.2 Where the universal adapter is needed.

This is not just a current fashion. The old way of building broadcast systems will not survive or return. This is because the old ways cannot handle the complexity, the interoperability, the development process, and the endless possibilities to a sufficient level.

But 'stay tuned' - solutions are on their way

1.2 Key concepts

We are used to working with isolated systems, interfaced by *people*. These are systems in *co-existence*. Today this is changing.

The demand to increase output and/or decrease costs calls for the optimisation of our processes and their supporting technology. As developments shift from traditional, dedicated, broadcast systems to IT-based solutions, isolated systems are being replaced by modular combinations of functions and data. To make this work, some sort of glue is needed to connect them all together.

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 optimise workflow.

The problems

When integrating systems using IT technologies the following questions need to be addressed:

- How does each system work?
- How to connect the different systems?
- How to find material and where it is managed?
- How to prevent re-entry of contents?
- How to ensure consistent versions of data across systems?
- How to co-ordinate events in the systems?
- How to avoid working with many different user interfaces?
- How to automate workflows across systems?
- How to provide seamless evolution of separate systems as well as the whole portfolio?

Introducing layers

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

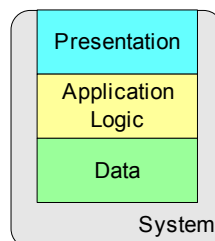


Figure 1.3 A layered system

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*.

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' is 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 beneath the databases and file-servers, taking into account the fact that physical storage can differ. This then allows you to change or scale the storage without changing the data layer. This fourth layer would be:

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

In fact, for more detailed analysis, these layers can be further sub-divided, or further infrastructure components added. For example, transport (different networks) and workflow. Also, layers are sometimes combined to create specific types of systems.

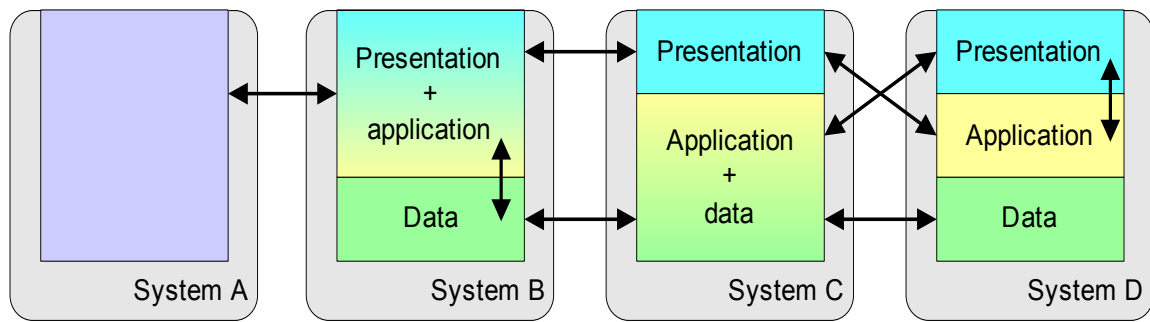


Figure 1.4 Examples of different 'layer' configurations. The arrows give examples of communication between layers within and between systems (see below on 'open or closed systems').

Figure 1.4 shows different layering combinations, ranging from a non-layered system (A) to a three-layer system (D). System B is commonly called a *thick client*; it has strong connections between the presentation layer and the logic-layer. System C has a strong and closed connection to the data layer, but a separate (for example web-based) client on top of it.

Comparing vertical and horizontal systems

Vertical systems	
Benefits	Drawbacks
systems individually visible & manageable	single point of failure: one brings down all
no cross-dependencies stand-alone independence	duplication of functions (= duplication of costs)
	complicated system evolution/upgrading as you have to keep all systems aligned
minimal message exchange requires lower performance network	likely based on proprietary/incompatible systems
simple redundancy strategy: duplicate device	
simple failure management: replace device	
	restricted/limited points of access when integrating with other systems

Table 1.1 Benefits & drawbacks of vertical systems

Horizontal systems	
Benefits	Drawbacks
distributed organisational responsibilities right responsibility at the right place	difficult to identify & contain failures
high flexibility & reliability of services e.g. on demand addition of servants for a service	high-performance network required
minimal waste of resources	distributed management & control = inefficient in critical situations
smooth system evolution due to modular architecture	organisation-wide accessibility may be expensive
simple & focussed scalability e.g. only develop the functions you need	requires much system integration
simple to integrate segmented innovations only need to upgrade a module	
	more complex to obtain redundancy
potentially many points of access for integration	these points of access might use proprietary protocols

Table 1.2 Benefits & drawbacks of horizontal systems

Both vertical and horizontal systems have the potential to address your requirements.

Open or closed systems

It is important to realise that there might be no connection between the actual construction of a system and how it appears or reacts to the outside. A system using the latest 'horizontal' technology² can still appear as a closed

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

system. The constraints may be imposed by unpublished or licensed interfaces, or by proprietary additions to known application interfaces (APIs), protocols, etc.

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

In the real world, we are dealing with a complex mixture of scenarios, and the solutions will 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 it is useful to treat separately.

We can go one step further and describe systems as component³ based.

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

These components can be grouped within the previously mentioned layers, although some components work across layers as well. As long as they keep the rule in the above definition, this should not be a problem.

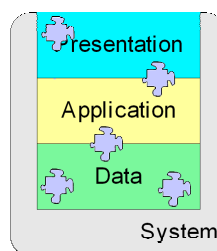


Figure 1.5 A components based system

Some of the components' interfaces should be openly accessible, though not necessarily all. This can be achieved in many ways and is one area where the need for standardisation arises.

The main point to remember is that systems should be component based.

1.3 Integration: How to 'Glue'?

The terms vertical and horizontal integration are used. It should be noted that these are different concepts to vertical and horizontal systems. See chapter 2 for detailed explanations

Middleware: the glue

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

One of the early techniques of "providing glue" was through the use of the data-broker (a hub and some adapters to the systems). Without going into technical details the concept and expected benefits are illustrated in Figure 1.6.

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.

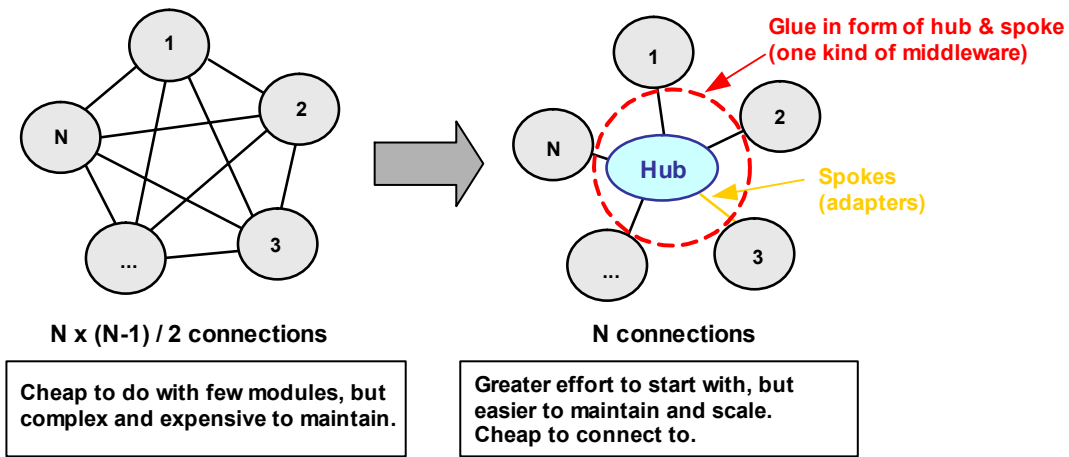


Figure 1.6 Example of glue

Here, the glue takes the form of a hub and spokes, replacing the one-to-each integrations between systems or modules, thus requiring fewer interconnections and replacing 'chaos' with efficiency.

Figure 1.7 combines the concept of middleware (glue) with the layers introduced earlier.

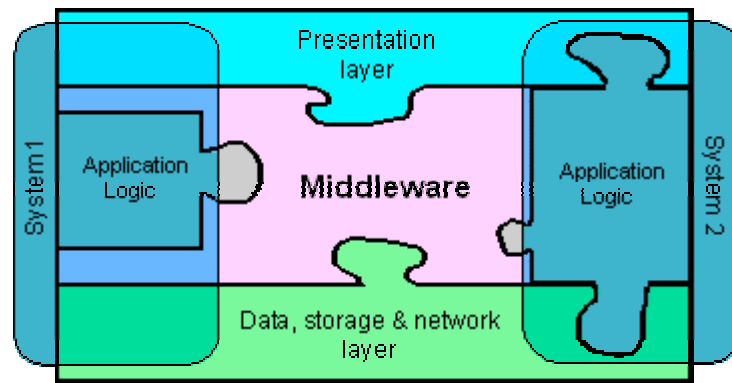


Figure 1.7 System concept, including interfaces⁴

The figure shows:

- The *top, or presentation, layer*, which represents the interface seen by the users.
- The *application logic*, which represents the functions carried out by the system.
- The *lower layer(s)*, which represent the *data, storage and network*.
- The *systems*, which use the logic to perform operations on the data and present the results to the user(s). This requires:
 - *Middleware*, which is 'the bits in between', allowing the layers and systems to interconnect and form a working system together.
 - Some applications (systems) might only allow one common interface to the rest of the world (right-hand side). Others might give access to middleware from interfaces in each layer (left-hand side)

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 the 'puzzle' pieces in Figure 1.7.

It is important that the interfaces allow for communication in a common, flexible, structured, efficient and re-useable way. We will return to that in more detail later.

Where do components fit into this?

As we discussed before, nowadays systems are more and more component-based. Figure 1.8 extends our layered model with this notion.

⁴ Acknowledgement: the concept for this figure is based on a figure by Bob Edge (Thomson GVG).

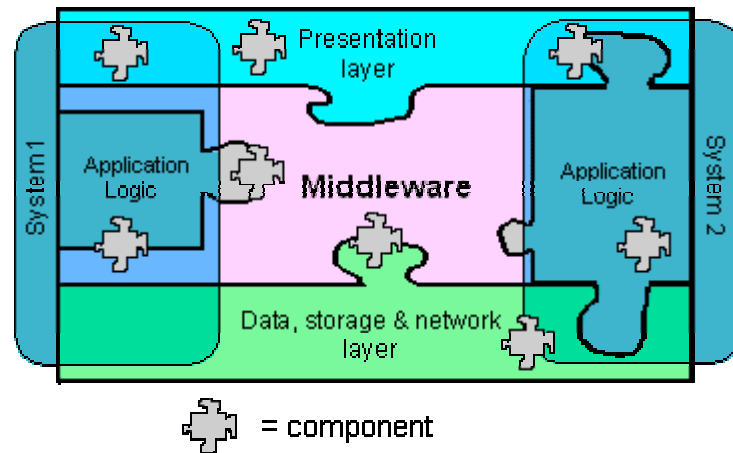


Figure 1.8 Systems can be built out of many small components.

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 to another kind of glue - another integration technique: Web-services. A Web-service is a technology that includes a mechanism explaining how and where to use it. The service could address both data and functionality. The description is done in a formal way offering the access in a total open and standardised interface. This makes, in theory, all integrations possible between each service and between other systems. The re-use is ensured by catalogues of services in your company and public catalogues where companies can decide to share some of their services.

Both brokers and Web-services are explained later on. This allows the components to be interconnected as well.

1.4 Scope of this work

We have seen that middleware is many things and that there are many ways in which middleware could be implemented. So the goal today is not to try and standardise middleware, but to establish a system integration architecture; an overall understanding on what to integrate and how.

Main scope

For the purpose of this document middleware should be seen as a collection of technologies that 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, we set the scope of our work as:

System Integration Architectures

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

This more modular approach calls for new modelling tools to describe and develop functions, data structures, systems and their relationships.

Working in this more complex world, the need is growing to define business processes as well as business objects (representations of real-world objects, e.g. a programme-item or a video file) and their properties in a common way across systems.

There is also a need for a structured way to host and update information about the systems, modules, interfaces, data, standards, etc.

These requirements are also addressed within the scope of our work.

Scope of use: the value chain

Most broadcasters have a two-tier value chain comprising Services and Programmes, illustrated in Figure 1.9. System integration architectures and the different middleware technologies affect the processes in this chain as well as the supporting processes.

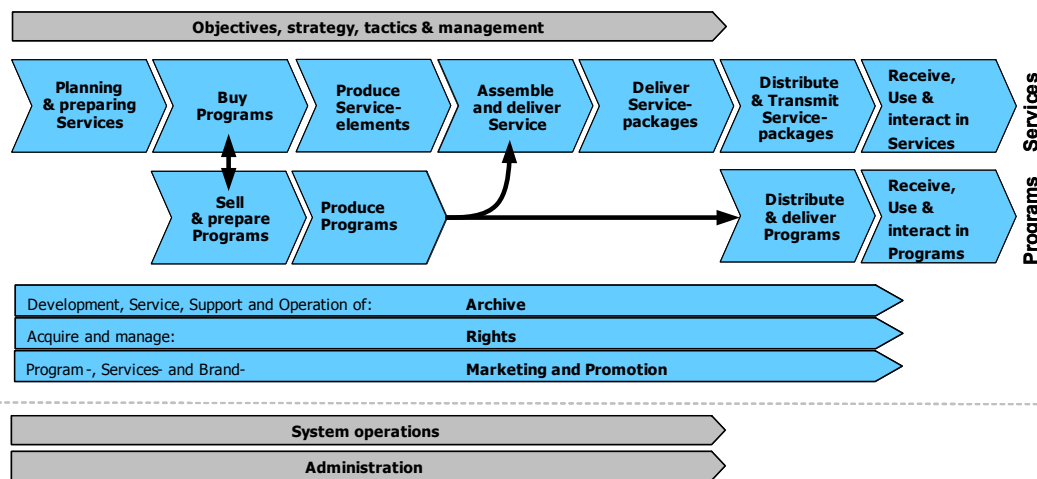


Figure 1.9 Typical broadcaster's value chain (source: DR)

The integration architecture ensures that information and functions are accessible by 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).

Broadcast specific scope

We can benefit from integration techniques commonly used in other industries in many parts of our operations. 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 form of text and numbers.

Of special interest for broadcasters should be the areas around:

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 realised 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 that relate to the processes in the whole value chain across systems or modules of functionality.

In the framework of the P/MDP group, it has not been possible to cover all of these areas. However, they have been used as guidelines and this document uses one of them as an example: the Ingest process.

The P/MDP group recommends that each of the four areas should be a subject of future work.

1.5 What is so great about a System Integration Architecture?

The arguments for your organisation to establish and maintain a system integration architecture can be summarised as follows (in no order of priority):

- **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 help to provide this flexibility.
- **Speed:** where functions and data have to be used across systems, middleware speeds development and changes (in the long term, though not necessarily in the short term).
- **Cost:** the costs of building and maintaining unique one-to-one integrations between systems are 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 in that you only have to change the connection from the primary system to the hub. Middleware can act as a hub.

- **Standardisation:** 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 integrity of the data across systems and situations of use.

For horizontal systems, effort spent in making the evolution reliable, secure and robust can be concentrated in the service layers rather than distributed between single applications. For example: if you are using a single storage service for a distributed environment you have to define and deploy a single access, authentication and protection strategy only once; but if you have a separate storage service per system you have to repeat such activity for each one.

- **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 organisation gets a better overview; both due to the standardisation and to the needed repository of information about systems, functions, data, integrations, interfaces and Metadata. This opens up opportunities for stronger change management processes. More specifically, the ability to search within data across systems is improved.
- **System planning:** standardisation allows technical planning departments to use tools that 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 optimising the workflows.

Planning of complex broadcast-systems in future could be much more like “Lego®”. Originating from the sequence of operations, the required functionality is described in standardised modules and can be compiled as requested.

The result will be requirement specifications with a much-reduced potential, compared with today, for being ‘interpreted’ into something different than intended. The gap between vision and reality will shrink in the same way as the vocabulary between broadcast engineers and IT-developers will converge!

- **Training:** an understanding of the workflows in which the operators are involved will become more necessary in the future than today. Therefore it will be necessary, not only for dedicated technical training for a device or software-application, but also for education on understanding the whole production-chain.

This can be much simpler if the system architecture is based on generic and pre-defined processes rather than on proprietary structures. Time and cost in training can be reduced and the trained staff can be deployed very flexibly.

1.6 Examples

Two examples of what might be expected:

- You can search across all archive databases and raw material without having to address individual archive systems. The middleware makes sure all of the archive databases are searched and the results are combined.
- You can use a single terminal to see different machines. From your desktop you could use an editing tool, browse the archive, look into the play-out schedule and check the billing system.

Further examples from the Members are described briefly in section 1.11 and in detail in the appendices to the supplement. They are:

Project A: a project of the Bayerischer Rundfunk, which implements an IT-based play-out and production centre. It features the most vertically oriented system integration of the three examples. It encompasses a messaging-based system for system integration.

Project B: a project of Danish Radio, which, at the time of this report, is in the process of moving from vertically integrated to horizontally integrated system architecture. Standards and policies for system integration are established and both a broker and a concept for service-oriented integration are in operation. This is being done in close connection with the implementation of a DR Metadata standard. This project can be seen as a type in between project A and C, coming from a vertical legacy situation, but aiming at a horizontally integrated system environment.

Project C: a BBC research project that studies a completely horizontally integrated production. This project can be regarded as having the luxury of a 'green field' situation, enabling it to try to utilise the benefits of horizontal system integration to its fullest extent.

1.7 Most common integration types

System integration architecture depends on your particular business model (the products and operations) and your particular IT-environment.

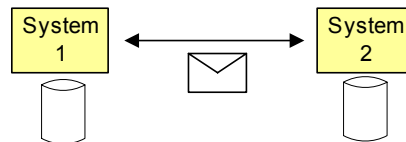
However, it is possible to describe some common basic components, to recommend tools, and to give guidance. It might even be possible to set up some standard requirements for the vendors of the systems we are buying, encouraging them to subscribe to an open and non-proprietary environment.

To start, we will now explain some commonly used terms.

Note that the integration architecture concepts we discuss are related to the planning, production, and play-out systems, as well as to 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 those described below are necessary. However, the management, control and reporting of this can be handled like any other kind of data in any other industry.

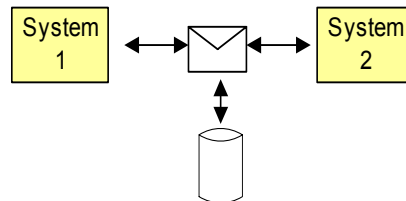
- **Message based integration** (copying data between systems)

The same data are 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.



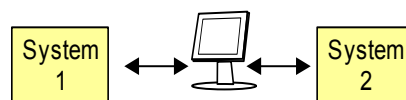
- **Data-carrying integration** (Different systems access the same data)

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, consisting primarily of an application server and a database.



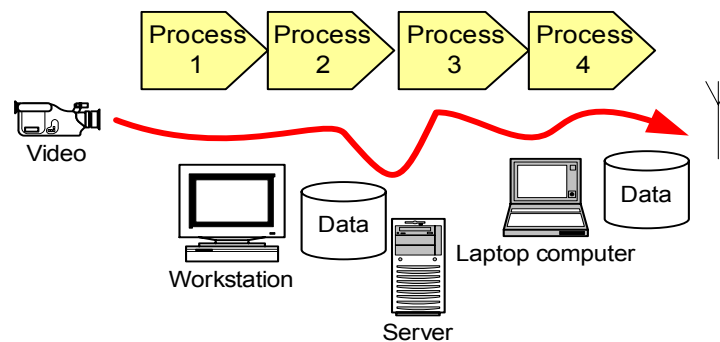
- **Portal integration** (Wrapping of multiple user interfaces into a single one)

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



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

Refers to automating or semi-automating 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. Work-flow mechanisms can support each of the three former described integration types.



Two main frameworks

In general, two technology frameworks have been successful in the market: MOM and SOA.

The earlier mentioned broker technology with a hub and adapters (spokes) to each system is the classical implementation of MOM.

Super-services, here primarily in the form of Web-services, are a major part of the SOA framework.

- **MOM** - Message Oriented Middleware

Message-Oriented Middleware (MOM) provides the abstraction of a message queue that can be accessed across a network. It is a generalisation of the well-known operating system construct - the mailbox. It is very flexible in how it can be configured, as it is based on the concept of software programs that deposit and retrieve messages from a given queue.

A message broker is application-to-application middleware, comprising one or more software facilities, and capable of one-to-many, many-to-one and many-to-many message distribution, using queuing, prioritising, confirmation, and other transaction handshakes. Other logical rules for the transactions might be added.

- **SOA** - Service Oriented Architecture

A service-oriented architecture (SOA) defines how two computing entities interact so as to enable one entity to perform a unit of work on behalf of the other. The unit of work is referred to as a service, and the service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other.

Simple Object Access Protocol (SOAP)-based Web services are becoming the most common implementation of SOA. However, there are non-Web-services implementations of SOA that provide similar benefits. The protocol independence of SOA means that different consumers can use services by communicating with the service in different ways. Ideally, there should be a management layer between the providers and consumers to ensure complete flexibility regarding implementation protocols.

1.8 Middleware services for broadcasters

Setting up broadcasting installations involves the solving of many 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 Service Oriented Architecture.

When using a Services Oriented Architecture (see paragraph 1.7), one might 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:

Understanding services

A service is basically a special system component that can be invoked by other components wanting a certain task to be performed on certain data. An example service is an authentication service, which authenticates (i.e., confirms identity of) a user based on his/her user name and password. In order to be able to invoke a service, it is necessary to know what service operations are available and how to ask for these operations. The specification of these is called a *service interface specification*.

How should 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 (supplement, chapter 3).

A useful strategy in designing a service architecture is to distinguish between services specific to your industry, the *broadcast domain*, and those not specific to any particular industry, the latter being commonly 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 that the users know best.

Pervasive Services

Pervasive services provide functionality that 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:

- Authentication
- Session Management
- Time References
- Workflow Management

For the pervasive services the users should select available services and implementation technologies that 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 that they meet the users' specific operating conditions.

Specifying services

For each service the following should be specified:

- General description
- Purpose
- Interfaces
- Dependencies

General description

The general description of a service should provide a detailed description of what the service is intended to be used for and why is it needed. For example: "The Transcription service provides speech-to-text transcription facilities to the other components of the system."

Purpose

The purpose of a service is a short description of the functionality it offers. Basically it is a more abstract description of the interfaces. Example: "The Transcription service receives requests for transcription and notifies the requester at the completion of the job."

Interfaces

Interfaces provide access to the functionality of a service. Depending on the implementation context, interfaces can also be called functions, methods, actions or operations. Interfaces include the parameters passed to, and

received from, the services when they invoke the interface methods. Interfaces also have behaviour; which is the dynamic characterisation of the interface. An example:

Transcribe([in] **fileURL** sourcefile, [out] **fileURL** targetFile, [in] **transcriptionOptions options**)

fileURL sourceFile: *file containing the audio track to be transcribed*

fileURL targetFile: *destination file for transcribed text*

transcriptionOptions options: *dictionary to be used, text format and encoding, etc.*

Other examples are provided in appendix D.

A framework for describing the functionality of web-services and the way to access them is available and known as UDDI.

Dependencies

A service may use other services, when required, for fulfilling its purpose. The knowledge about the dependencies on other services is very important for developers and integrators. For example: "The Transcription service depends on the File Manager service."

Different Views

There are different types of players involved in the development of IT-based TV programme production systems: typically-

- Planning engineers
- System integrators
- System developers

These different roles require different views of the system that is being modelled. For this reason we propose to include three different views in the specification of the service architecture.

- 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.

A more detailed description is provided in chapter 3.

The P/MDP group recommends that this should be further investigated.

1.9 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

Arguably the best-known modelling methodology is UML (Unified Modelling Language). In the last 10 years it has established itself as the predominant formal notation for the description of IT projects.

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 that describes the process of recording with a VTR.

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.

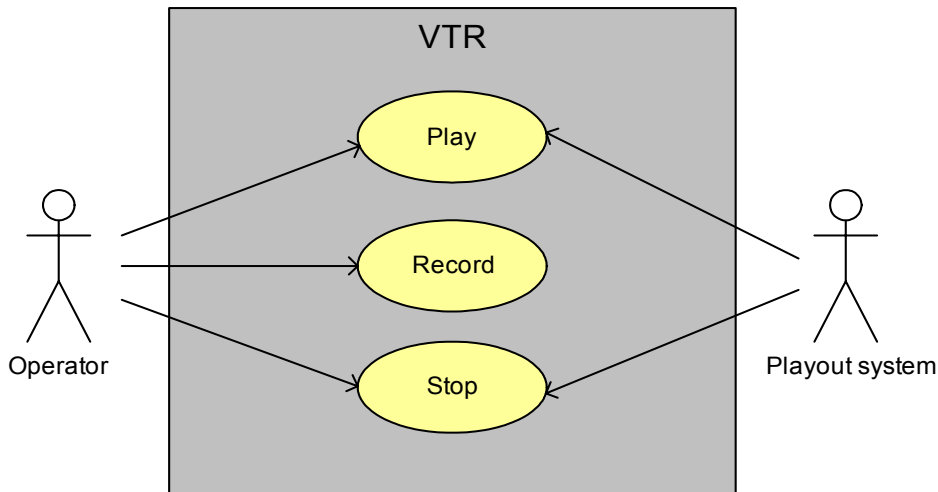


Figure 1.10 Use case scenario for use of a video system

Class diagram

A Class Diagram gives an overview of the structure of a piece of software. Classes are the declaration and implementation of real world objects in a programming language (e.g. C++, Java, etc.). Usually the Class Diagram shows the classes and the associations between the classes.

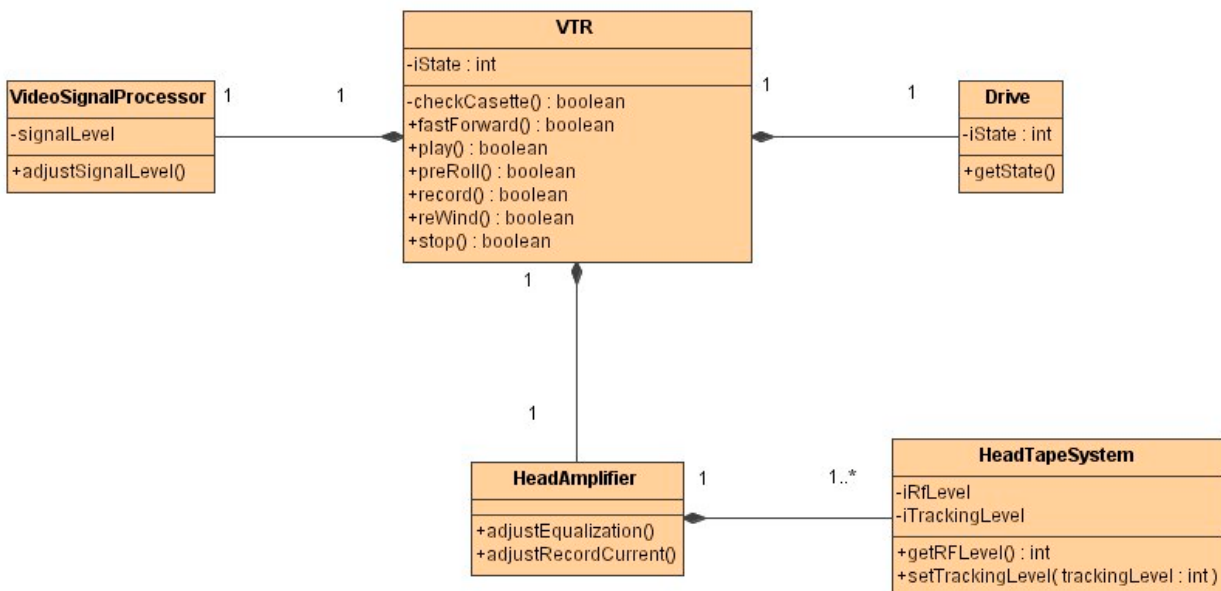


Figure 1.11 Class diagram for the VTR example

Activity diagram

In an Activity Diagram you can see a roadmap for sequences of activities and decisions. It also shows you the states and behaviours of the system.

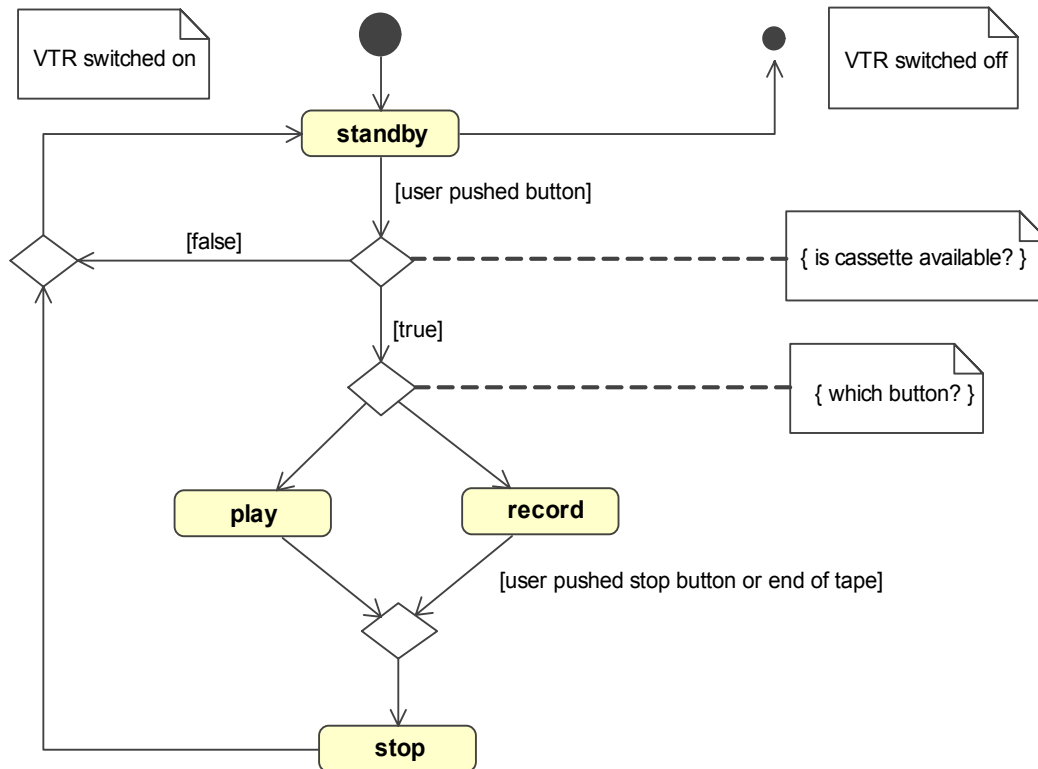


Figure 1.12 Activity diagram defining behavioural aspects

Modelling frameworks

Frameworks exist which give some guidance in how to deal with system analysis and description.

- The Model Driven Architecture (MDA)
- The Zachmann framework

MDA is an architectural approach that splits off application domain and implementation platform issues. To achieve this, MDA uses the concepts of the Platform Independent Model (PIM) and the Platform Specific Model (PSM). The PIM as well as the PSM can be specified using UML. The Object Management Group (OMG) has developed MDA. This is the same group that standardised UML.

The MDA approach is useful as a declarative concept and may be used to separate different concerns and meet the appropriate level of abstraction. It may also be useful to set the scope of standardisation efforts.

The Zachmann framework has a wider scope of system description than MDA. The idea is to make a structured shortcut to the descriptions and description methods relevant for some pre-set and particular use. The pre-sets are defined in two dimensions:

1. What, how, where, etc.
2. The level or functions for each of these: managers, process designers, system developers, etc.

For an illustration and references refer to annex E (supplement).

1.10 Experiences from the broadcast world

1.10.1 Project A Bayerische Rundfunk (BR)

How middleware was intended to be used

When BR started planning SZFM in 1999 we did not dare think about “middleware” in terms of linking together the storage-layer with all of the applications in the upper layers. Yes, there were some pieces of software offered by broadcast vendors, which they called “middleware” but they did not have the generic approach we are talking about now – nor did they fulfil the requirements for a distributed service based architecture.

Nevertheless, some systems, like the CMS, did provide service based structures, but only internally – not accessible from “outer-space”.

So we had to take all those encapsulated systems and stick them together with the glue of APIs and proprietary protocols with all their advantages and disadvantages and with the doubtful hope that we would never have the need to replace one of our “babies”.

Positive aspects

Not using “middleware” at that point of time was a positive, because there was none. In fact, there were signs of its emergence but the offers we got were not affordable and the reliability of the whole system seemed very doubtful. Particularly for the latter reason we changed to a more “vertical” solution, giving us the confidence that the play-out would actually happen, every day.

Negative aspects

The advantages of closed and therefore stable systems are won at the price of a serious lack of interoperability. One example is our SAN-based editing system. It proves very difficult to find a solution for transferring files to and from the system. The system vendor only offers a collection of indifferently documented SDKs. Consequently, each manufacturer which has to interact with this system needs to develop their own implementation. This means programming a lot of NLE specific software code.

Another challenge, in the context of restoring archive-material, is the linking between the imported high-resolution material, e.g. from a file based archive, and the associated Edit-Decision-List (EDL) generated from the content-management-system via browsing.

Our recommendations for future projects

Test as much as possible before placing an order, insist on standards, do a thorough job in analysing your workflows and always be sceptical about interoperability promises!

See annex A (supplement) for a detailed description of this project.

1.10.2 Project B Danish Radio (DR)

How middleware was intended to be used

As part of a larger corporate strategic development project the technological impact of the business strategic objectives was analysed in 1999 and 2000. One conclusion was the need to move from closed vertical systems into horizontal, n-layered, system architectures.

Research into available technologies, market trends and offers as well as structured analysis of the portfolio of systems in DR was carried out during late 2001 and 2002. Major opportunities for better system integration were described, and a handful of the most important were chosen. Some of them were picked out for ‘proof of concept’.

The final output was ‘Standards and policies for system integration in DR’ and a recommendation for the purchase and implementation of a data-broker as the first move.

The broker, as well as major parts of the policies, was in operation in spring 2003. A revised version of the standards and policies is underway, based on the actual implementation and the first experiences. Particularly the strengthened focus on Service Oriented Architecture, which was just arriving as a challenger to broker-based integration architecture in the early days of this project, is now taken into account.

Besides the importance of the technology, the project is, at the same time, seen as a change management programme, as the process takes a considerable amount of time, and the rebuild of skills and development procedures are a vital part of making this change successful.

Positive experiences

The new system integration architecture is the basis of several major technology projects. It is a positive experience to be able to handle these new projects in a new and ‘right from the start’ integrated manner. The already-running server based production of TV news and the coming roll-out in the rest of the TV production would not have been possible without this foundation. Several projects around administrative workflows also benefit from the broker-methodology in operation.

This experience was a distinct advantage when meeting the vendors of the EU-tenders for our new broadcast house.

Speed and flexibility in making system integrations are definitely achieved, and some new products have even showed up as cheap side effects.

Challenges

Establishing technology, skills and procedures for a new system integration architecture is definitely a long-term project and nor is it trivial. It takes a long time to change old development routines to take this new architecture into account as a fundamental way of working. A strong management and some financial control and buffer mechanisms are needed to deal with the new balance between maximising general benefits across systems and projects, opposed to maximising the independent project. The first projects that need integration, so that other will prosper later on, will have a different cost profile than the projects succeeding them.

Some vendors are still trying to fence-in customers by having a closed and proprietary way of dealing with integration with other systems. They want to own the world and the way to address and control other elements in that world. It takes time to make vendors accept that we will not necessarily buy the whole of their package. Moving from 'one fits all' for minor and mid-range broadcast stations to a 'best of breed' for larger stations means great differences in the approaches from the vendors with regards to the needed openness of their system architectures.

Recommendations for future projects

Now that broker technology has become more widely spread as a kernel of some larger applications in the broadcast world, so broker-to-broker integration is more and more common. Analysis of performance issues, including the need for real-time performance in a broadcast world, should be carried out. The same applies for applications working across asynchronous and isochronous networks (e.g. IP-network and real-time SDI and AES-EBU networks).

See annex B (supplement) for a detailed description of this project.

1.10.3 Project C BBC's DP (Desktop Production) project

Project Goals

This is a research project so the deliverables are not a working system, but a strategic architecture. The goal of the project is to identify and define the various standards, interfaces, architectures and generic technologies to which we should eventually move. The BBC currently has a number of projects implementing solutions that have short practical delivery time scales but inevitably incorporate commercial compromises.

The DP project operates by developing simple 'proof of principal' software demonstrators, but not in a form for operational use, which would require many times the resources for management, analysis, documentation, and rollout. Hence, most of the recommendations, while based on real software developments, are not products or live services.

Middleware

A key part of the proposed architecture is that there is a middleware layer that links broadcast components to broadcast infrastructures. In fact, we have identified two middleware layers that could be different. We know that while there could be a common choice for all middleware technologies, in reality, the complexities of agreeing this with all providers mean that this will never happen in practice.

Workgroup Middleware is the high performance, media handling middleware within a relatively small workgroup. A workgroup is a number of seats, typically 2 to 30, where people are co-operating on the same production or using common media. An exception is news where the workgroup is typically much bigger.

Enterprise Middleware is the business handling middleware that links not only parts of a large enterprise, including the workgroup, but also major business functions like play-out, archiving, business Metadata, rights, accounting, scheduling and commissioning. This middleware links the whole organisation for common data handling, and infrequent media transfers. We expect it to be present on all networked seats. From the network point of view the enterprise middleware unites the network into one whole so that everyone has access to the same core services.

Workgroup Middleware

A co-operative group of production staff requires daily access to media files at high quality. This imposes a high performance demand on the middleware and its underlying network and storage technology.

See annex C (supplement) for a detailed description of this project.

1.11 The vendors' perspective

Types of vendors

Current-day broadcast system vendors form a heterogeneous group. Simplified, 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 strong tendency to favour 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 with IT standards nomenclature. The first can be deemed undesirable by its inherit 'lock-in' character, while the second may not prove any better, as there still could be 'data lock-in'. That is: the system may be open, but if you don't understand the data-model inside, you still cannot access it.

The good news is that the above represents the extremes and in practice both 'camps' are moving towards a better understanding of the customers' requirements (as are the customers themselves).

It is noticeable that it tends to be the smaller companies that have a better capacity for understanding the customers' particular requirements. It is likely that an open standard approach tends to suit smaller manufacturers' business requirements, as they are more likely to be able to supply process elements into an "end-to-end" broadcast structure. This will tend to increase the application choices users will have in the long term. However, some larger manufacturers also support an open standard approach to structural design as it could lower implementation costs. This approach should be encouraged.

Key observations

Vendors are opening up their systems

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

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 argued 'others should interface with us'.

Vendors wait for broadcasters to agree

Most vendors have no major incentive to standardise interfaces as long as broadcasters cannot agree on major definitions of business objects (what is a programme?, what is a service?, etc.). They will be happy to deliver all the different flavours of the same and all the different kind of glue or kit needed to make it fit together. They will hardly be the driver of a broadcast domain model.

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 that 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 remarked that the broadcast industry is special in that much R&D is done by the customers themselves, which is not the case in the traditional IT industry.

Middleware reliability is ok, but at a cost

In general the vendors did not see difficulties with middleware reliability, but they did warn it might 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 also other factors such as scalability and reliability).

While it is true that there is potentially a large amount of development associated with producing reliable middleware, this should be balanced long term by system flexibility and the reusability of software elements within products as interface standards emerge. The general message seems to be that a mixed environment will have to exist for a while.

1.12 Managing System Integration

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

To boost economy, quality, capability, etc there is a need for standardisation and streamlining of the integration efforts. It is important to have a strategic scope that spans more than just the current projects. If possible, this responsibility and governance of system integration should be centralised within the company. The consultant company Gartner has named such a function - Integration Competency Centre, ICC. This should be complemented by appropriate management and control mechanisms in the business units and at corporate level.

A general overview to help allocate responsibility is given below. This figure applies regardless of whether the vendor or supplier is internal or external:

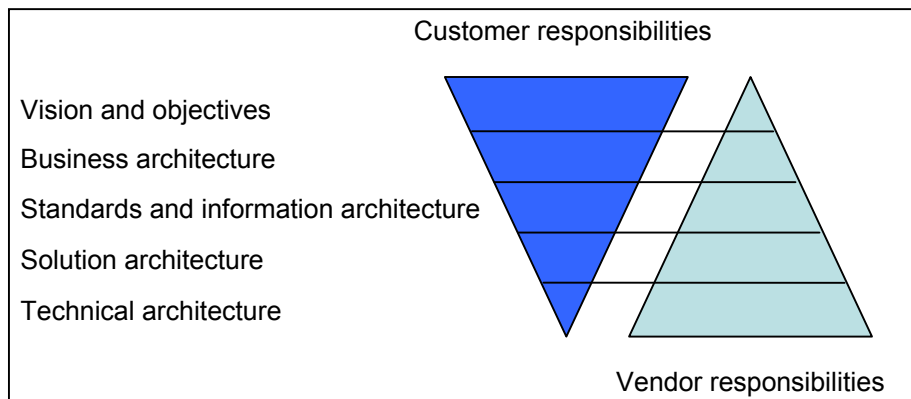


Figure 1.13 Who will be responsible for what?

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

Describe what you do and do what you describe.

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

Some guidance and first steps are given in chapter 5 (supplement).

1.13 Standardisation

There are two areas where standardisation efforts would be welcome:

1. A common (core) Metadata model

There is a lack of a common (core) Metadata model. This is the Priority One 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, scheduling, are the broadcasters' 'bread & butter' services. It seems that standardisation 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. The effort should start with words & models. Once the services are defined, it should be investigated whether toolkits, APIs, etc., would be of value. Where standards are immature or ambiguous, reference implementations would be an advantage.

Who should standardise these services?

A combination of an IT organisation (e.g. OMG), together with a broadcast organisation (e.g. EBU or SMPTE) would be preferable. There are also standardisation bodies such as the IEC who are useful as they work in both the media and IT fields. Some manufacturers noted that they would like to see a community effort, e.g. an

open forum on interoperability, using more Internet-based tools and less physical meetings, which would also allow smaller parties to be involved.

When should it be standardised?

- It seems that currently some manufacturers see little urgency for standardising middleware related topics. Many of them have just started to implement file-based technologies, such as AAF/MXF support in their products, and they need time to achieve sufficient Return On Investment.

- Middleware elements have already been around for more than ten years and, as one manufacturer put it: 'there is no immediate penalty if you don't use them'. However, although the middleware that has been in use may offer technical solutions, the semantics for the broadcast & production industry are still in their infancy.

But-

- In view of the developing convergence of traditional broadcasting techniques with IT techniques, and also that any development today has a strong business element, broadcasters now need the ability to specify IT based applications and business structures within their organisations. It is now a matter of urgency that standardisation work begins, to enable users to specify functionality between products.

- Taking into account that standardisation takes time and the use of middleware is clearly a common factor in other (influencing) industries, starting as soon as possible seems the wisest course.

1.14 Requirements

Many requirements can be derived from the Golden Rules in chapter 9 (supplement). We have tried to provide an outline of requirements for use with vendors in chapter 6 (supplement), but as this is very dependant on your own environment, you should build further on this work when going into detail.

1.15 Golden Rules

The experiences shared by the P/MDP Members resulted in a list of 'Golden Rules' for broadcasters, vendors and standards organisations. Ten are listed below, see chapter 9 (supplement) for the others.

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

Table 1.3 Ten Golden Rules for system integration

1.16 Future opportunities

1.16.1 The transport layer

Dealing with real-time rich multimedia, we as broadcasters have obvious needs for high performance transport of content. Regardless whether we are talking about setting up requirements for transactions in middleware or simple transfer of large media files, this is a large area to be covered by future work.

1.16.2 Grid computing

As described by Ian Foster and Carl Kesselman [1]:

“A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computing. **Grid computing** is a hardware and software infrastructure that clusters and integrates high-end computers, networks, databases and scientific instruments from multiple sources to form a virtual supercomputer on which users can work collaboratively.”

A Grid structure requires two key elements before services can be provided. These are high bandwidth network interconnections and Grid middleware to facilitate grid applications and services.

Which benefits to utilise, and how they can be applied in the broadcast world should be investigated in the future. Some descriptions are provided in Annex F (supplement).

1.16.3 Meta-Metadata

Typical tasks that middleware components have to solve are:

- The adaptation of the meaning and structure of the I/O parameters passed via their interfaces.
- Adaptation and extension of their own behaviour, to achieve the new goals of the integrated system.

To allow both of these to be generalised, the concept of 'meta-Metadata' may be useful. This is currently being researched by various projects. See appendix F (supplement) for more details.

1.17 Glossary

This annex provides definitions and examples of commonly used terms.

Agent (software agent)

A software program that performs information gathering, information filtering, and/or meditation on behalf of a person or entity.

API, Application Programming Interface

The calling conventions (interface) by which a computer programme communicates with another computer program.

Asynchronous process

A process that operates independently of other processes.

Broker

An entity used to exchange data between systems with a guarantee of integrity.

Compare with a Stock Exchange trader.

Component

A component is a 'black box' with known and described interfaces that can be used without knowing anything about its internal structure or internal functionality.

Framework

The underlying structure of something, e.g. a software system.

It may consist of service and interface specifications.

Interface

A shared boundary between two entities (processes, modules, humans, ...) offering a point of communication between the two.

Message exchange (concept)

Method for communication of structured information between software components or applications.

Middleware

Software used for coupling high-level system components (applications and user interfaces) with basic system components (network and data storage components).

Object brokerage

Can have different meanings, depending on context, but essentially the technology used to share (distributed) objects.

Portal

Used in many ways. One usage is as a reference to web-portals acting as a container for many kinds of content or functions. But it is also used for the special technology that integrates various components at a portal server.

Real-time

A characteristic of systems that respond to stimuli within a small upper limit of response time (typically milli- or micro-seconds).

Role

An identifier for behaviour and response of somebody or something in a specific activity/scenario.

Service

A service is a software agent that performs some well-defined operation (i.e., “provides a service”) and which can be invoked outside of the context of a larger application.

Super Service

A general term. Combination of services in order to perform a more complex task.

E.g. end-to-end workflow. The most widely spread type is called Web-services.

Synchronous process

(Not the meaning of ‘synchronous’ as used in the video world).

Refers to a process that is dependent on other processes.

In a synchronous communication a service requester has to wait for a reply from an invoked service before it can proceed

UDDI

UDDI (Universal Description, Discovery, and Integration) is an XML-based registry for businesses world-wide to list themselves on the Internet. Its ultimate goal is to streamline online transactions by enabling companies to find each other on the web and make their systems interoperable for e-commerce.

UDDI is often compared to a telephone book’s yellow pages. It allows businesses to list themselves by name, product, location, or the Web-services they offer. Each company can have an internal UDDI catalogue. The services exposed to the world outside the firewall can be a subset of this.

Use case

Use cases are the UML modelling technique for formalising the functional requirements placed on systems. Use cases do not describe the internal behaviour of a system.

Web-services

A Web-service is a technology that includes a mechanism explaining how and where to use it. The service could address both data and functionality. A formal description of the web-service allows for access via a totally open and standardised interface. This in theory makes integration possible between all services and (other) systems.

Re-use is ensured using catalogues of services. For example a catalogue of services in your company or public catalogues where companies can decide to share some of their services.

-----End of document-----