

## Creating an MLE

### MLE design

- Design inputs
- Design criteria
- Design process
- Design outputs
- Design tools and techniques
- Domain models
- Architecture
- User interaction design

## 6. MLE Design

### Who should read this:

people who have been charged with delivering an MLE to an organization. It will also be of significant value to a project manager wanting to acquaint their development teams with the key differences from other systems development and design approaches.

### Outcomes:

On completion you should:

- have an awareness of the key issues and concerns for the designer,
- be able to outline a flexible design process,
- be able to identify key tools and approaches, and
- be able to identify key deliverables

### Approach:

This section aims to provide you with a set of concepts and tests that can be applied when designing an MLE. These concepts and tests have been derived from a wide range of experience in the design of MLEs and related systems.

### Introduction

The process of design in IT terms is very often treated as an engineering problem much in the same way that a bridge might be built. The idea of having one goal right at the start does not fit into the reality of MLE design. This fixed goal construct can lead to a number of problems in the design of a MLE.

The major problem that you will have when designing a MLE is keeping an eye on the ball. The MLE is not a single thing; it is a delicate and often conflicting web of technical, social and political issues which have to be resolved to an acceptable level. In very few circumstances will a single technical solution actually deliver a useful product for your institution.

The development of a MLE is like a very long term plate spinning exercise. The trick is to get all the plates spinning and keep them spinning throughout the development process. All elements of MLE development are linked together and a successful design is a product of successful implementation of all the other stages both before and after. It is more useful to think of your project in layers that interact rather than solely in stages. This is the inverse of the waterfall model where stage 'a' follows stage 'b' and more difficult to explain and model.

As we explore the design process only the key interactions will be highlighted, you must remember that all the plates must be kept spinning and that the process is highly individual. The amount of effort you need to put into keeping each plate spinning will vary for each institution depending on a wide variety of factors. The level of organisational integration will have a big impact – if the institution is highly centralised then some plates will need more attention, if the institution is highly distributed, again a different balance of effort is required. The other top level thing to bear in mind is that there is no one magic technical bullet to solve the problems, as each institution is different.

The design process itself needs to be constantly in a state of change. The design of an MLE needs to reflect the user needs, and institutions never stand still. Even though we talk about design, this means different things to different people; and there are a number of elements you need to consider in thinking about it.

Firstly, what is in a design? The answer is that there is never only one design but a whole raft of different views of the architectural whole. In the world of architecture, for a new house, we could expect to see an overview for the clients, a plumbing diagram for the plumbers, a wiring diagram for the electricians and so on. The problem with an MLE is that it is not one single thing but a series of agreements between systems, all of which may be evolving at the same time.

In this section we will look at different ways to manage the design process to generate the required outputs to drive the development processes. This may sound like a linear process but the truth is that design, build and the other stages are constantly being reintegrated as new pressures emerge and are resolved, so design changes user expectation, which changes other systems and so it carries on.

These core topics provide a high-level view of the design process, from sources of input through to deliverables:

#### *Design Inputs*

This is an overview of the inputs to design – organisational, technical, user and so on.

#### *Design Criteria*

This section covers some of the core criteria for MLE design, including usability, user requirements, security, and some of the constraints on design such as legacy systems and legal issues.

#### *Design Process*

This section looks at some of the broader design issues such as architecture, interfaces, and design methodologies.

#### *Design Outputs*

This section looks at the deliverables of the design process, such as specifications and documentation.

The following topics cover some specific areas of design important for MLEs in more detail, including practical advice, and resources such as design patterns that can be applied to a number of situations.

#### *Tools and techniques*

This section looks at tools that can assist in the design process, including the use of the Unified Modelling Language for diagramming.

#### *Domain Models*

This section provides an overview of how the outputs of the requirements process are translated into models that can inform implementation.

#### *Architecture*

This section provides an overview of some common patterns for technical architecture, which can be used as the design template for the architecture of an MLE or be the focus of discussion about the overall technical direction of an MLE design.

#### *Interaction Design*

This section covers the design of the interactions between users and the MLE, including usability, labelling, navigation and structuring issues.

## **6.1. Design inputs**

The design process has a number of inputs from different stakeholders, the organisational profile you have been building will help to identify where those inputs are likely to come from. A good starting point for designing the MLE is to look at the plates already spinning to see what we can gather into the design. The design needs to be continually refined to ensure that it satisfies all the stakeholders' objectives over time.

The area of design inputs and outputs needs to be considered from three perspectives, organisational, technical and user.

### **6.1.1. Organisational**

The organisational drivers for a design may impact at all levels of the design. It is in this area that the profile of your institution becomes very valuable. There may be a set of ownership issues with the data providers. The identification of what data sources exist and how they might be used, is a key input as this may be setting the limits in a number of other areas. The actual structure of your organisation is a good indicator of the types of issues you may face. As a rule of thumb the more distributed the organisation, and by that the number of autonomous systems it comprises of, the less likely you are to spot any organisation wide errors or problems. Conversely in a highly centralised organisation you may spot significant holes in systems though there are a smaller number of systems to actually integrate. It helps to conceptualise the organisation in a number of ways.

Both the GIMIS and INSIDE JISC projects are worth comparing for organisational culture, the first was delivered to a highly centralised organisation, the second integrated a wide range of systems from a highly distributed institution.

In many cases it can help to look at the different ways in which the organisation may be modelled, as this often identifies different key stakeholders and agendas. In 'Images of Organisation' Morgan articulates a number of different models it is worth considering, and Schoen in 'The Reflective Practitioner' provides different ways to think about the investigation of your specific environment. The area of requirements gathering also offers a lot of information on the socio-technical issues.

### **6.1.2. Technical**

At the technical level it is clear that there are a number of different technologies and processes which could be used effectively. This may also be the domain of other systems specialists. The involvement of other technical users is often a 'religious' issue, if you choose a development, tool or language which is not the same as another expert group then there is likely to be some tension. There may be very good reasons why you choose a particular technology but, just be aware at this stage in the design, that this is where lines can start getting drawn.

The heart of the matter in this area is often about self image, this might seem strange from a technical perspective, but just as other ideological beliefs, such as political affiliation, separate people, technology can play the same role. If someone has always been accustomed to a particular approach, or to a technology supplier e.g. Microsoft, they will have invested a lot of personal time and effort in learning those technologies to build a skill set and a professional image. If someone else then comes along and says 'Unix is better', this can be very threatening. It directly calls into question the fundamental platform that the person's professional ability, and therefore self-image, are based on. There is no single right answer, as all technologies just have different attributes, each has strengths and weaknesses. The key message is, do not assume that your perspective on the applicability of a given technology is shared or that it is fixed. Bearing that in mind and ensuring you are sensitive to these issues, will enable you to reap significant benefits from your MLE.

Pre-existing systems often exist in organisations, even though these may not actually be called an MLE. 'The student web database' may well be a prototype MLE, and as such, sensitivities may be well be present when your project is looking to replace the system. Tread gently with technical staff as often they have a particular view of the world which is not strategic but highly technical and highly creative in that environment. You may well be looking to replace a system that has taken ten years of their lives to produce. Do not expect such a major change to be stress free for an individual.

### **6.1.3. User**

The user has already done a significant amount of work as part of the requirements gathering exercise but in addition it is often worth going back to the users to check details and gather comments, especially when it comes to areas where compromises need to be made. The user is the ultimate customer and

the MLE you produce must bring real value to them or it is just a waste of money.

One problem is that we often talk about 'the user' in the abstract, however the diversity of users for an MLE is significant and you need to draw upon previous stages in the development to ensure that the design reflects these diverse and occasionally diverging needs. The actual resolution of conflict in the design will happen as the design migrates based on the iterations of discovery and testing. This is one area where each time this stage is revisited, to 'keep the plate spinning', that additional effort needs to be applied to identify and mitigate both risk and conflict.

As an analogy, consider the development of video camera technology. In the first generation the user had never seen a video camera and was happy to be able to capture footage and play it back on their TV. As experience and familiarity with a system grows then the user starts to demand more, a lighter weight camera, longer battery life etc. So the next iteration takes place. Now this first iteration could only deliver one of the above requested features as they were mutually exclusive, a lighter weight and a longer battery life. At this point you have two groups of users who differ about the most important development direction and so the tension rises. The analogy was designed to be simple but in your MLE the issues are likely to be far more complex without clear boundaries. This is the process where constant references to the other layers will pay off in building the big picture.

#### **6.1.4. Databases**

For a successful MLE implementation it is essential that the data sources involved, often databases, are well understood. The structure of the database and documentation of the interfaces is essential as often you will want to pull partial data out of the database for aggregation. Where there has been in-house development this should be easier. For some suppliers the structure of the database may well be one of their most closely guarded secrets. This is for two reasons, firstly there will often be significant intellectual property in the database and it would provide a significant advantage to competitors to know how to integrate products. Secondly, the supplier is often concerned that other developers will inadvertently damage the database being integrated and destabilise their software while attempting integration.

In terms of risk, the least risk is from harvesting data and this is unlikely to cause a problem, unless it places a significant load on the machine. The next highest risk level is from updating the database at this point in the design as people may well start to get concerned, either because you may destabilise the primary application the database was designed for, or you may accidentally destroy real data. The highest risk for database integration in the design is where the design proposes additional data to be stored in the primary application. It is recommended that this is avoided. The chance of destabilising the original application is very high. Often programmers use shortcuts or knowledge of the database structure when writing code. If you add a field to a database those shortcuts and assumptions will become incorrect and may crash the system.

#### **6.1.5. Processes**

The design process itself has to encapsulate a number of other processes. The designer in this context is behaving more like an architect in that a number of different design options will need to be assessed, these will then be worked up into a set of plans, with different plans being produced for different audiences, and these plans will be iterated as issues are identified and resolved.

Each institution has a different set of processes but it is worth identifying some of the key ones and providing a simple rule of thumb for the design process. The key processes are again in the organisational, technical and user areas.

In the organisational arena there are likely to be some organisational committees or similar who will want to have sight of the design and their processes and timings need to be taken into consideration. There may be a number of different committees who will have to agree about various aspects of the

MLE, such as to agree the impact on the student experience in support of Teaching Quality or Audit requirements. There may be groups which oversee IT in your institution which again may need to be consulted.

There are also probably a significant number of business process which may be impacted by the design, and an understanding of the changes, an MLE will bring to working practices is vital. Follow this link to the infoKit on [Process Review](#).

If there are pre-existing technical partners they may need to be consulted and have the opportunity to bid for any IT development work before the work commences. This is common where there has been a move to facilities management of IT systems and services.

User processes are many and varied, you will have to try and integrate their meetings and agenda into the design. The students union is likely to be very interested in the project and may request representation on the project and their processes may need to also be considered.

### **6.1.6. Technical specifications**

As part of the design there are likely to be a number of technical specifications, designs and plans for existing IT solutions that need to be designed into ensure that return for effort is maximised. If the environment is complex there is likely to be a large number of existing systems. In the case of the INSIDE project, over 50 databases were identified for potential integration.

The diversity of technical specifications may mean compromises in the design due to issues of different technical strategies employed. For example if one database is based on Java and JDBC and another is based on Microsoft's .NET products there may be a choice in the design over which technology is going to be the main platform for the project. The decision is not actually made at this point, it will usually either be a constraint on the project or it will be decided just before the design is realised.

There is very little advice available in choosing between technologies and the answer is often provided by the environment. It is worth looking to the institutions information strategy to see if there are any clear guidelines on choice. All technical developers will have an opinion on what to use.

### **6.1.7. Standards and Protocols**

Andrew Tanenbaum is quoted as saying 'The best thing about standards is that there are so many to choose from' with regards to network protocols. A protocol is designed to provide a commonly understood and agreed way to conduct a process. These protocols govern many areas of our lives from the mundane, 'on which side does the knife go' when arranging a place setting on a table, to the highly technical protocols of how networks will operate.

Protocols can both help and hinder. In the design it is likely that a number of protocols will be considered for inclusion. These protocols can cover the business process, but are more likely to cover aspects of the interoperability between systems. These protocols are by their very nature not cutting edge. If a protocol exists, it usually exists on different vendors' equipment because it has been around for a while and by definition is not the latest greatest research technology.

The types of standards you may need to include in your thinking are:

- Database standards: ODBC, JDBC, Z39.50, SQL
- Programming Standards: Java, Visual Basic, C++, C, C#
- Network Protocols sets: TCP/IP, DECNET, IPX (Novell), AppleTalk
- File Sharing Standards: NFS (UNIX), ATF (Apple), SMB (Windows)
- Data interoperability standards: IMS, SCORM, MARC, RDF

There are numerous other standards, and all of the above list, are collections of standards rather than single standards. The key is to understand that where a pre-existing standard is in use in the design, there may be a requirement to integrate it and this needs careful consideration.

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.2. Design Criteria

In the first part of this plate spinning exercise we have gathered all the information and organisational context together to build a rich picture of the environment both human and technical. In Checklands book the first 2 chapters give another perspective on the problem of modelling situations and particularly about the role of the big picture. The next stage is to set some criteria that you need to achieve. It is easy to focus on the functionality of the design at this stage, and while functionality is vital, there are a number of other criteria which need to be considered. Getting the balance right is a key issue.

### 6.2.1. User requirements

The key area for design to focus on is the user requirement; does it do what the customer wants? Does/can the customer know what they want? The requirements gathering processes should be tracking these, and a well understood approach needs to be taken to reflect these requirements into the design and at the same time provide timely feedback to the users to identify areas where the costs may be too high or be simply impossible.

The process of ongoing feedback is the responsibility of the project manager and she should be planning a communications strategy to support that. In the case of the JISC project MARTINI they sent out monthly email bulletins to users keeping them up to date on progress.

The process of gathering user requirements is intimately involved with this and these should be integrated to deliver best value. See the section on '[Requirements Gathering](#)' for more.

### 6.2.2. Usability

An area often overlooked, is usability. Usability is the part of the design process that actually deals with what the user sees. This may be the selection of the taps in a bathroom in a building design or the user interface of your MLE. There are a number of well known approaches to ensuring the thing is usable and projects like the one at De Montford University took a very useful approach to delivering usability for students with a considerable amount of effort being put into the interface design. For an overview of the topic you should read Donald Normans' 'Psychology of everyday things' which provides a very accessible understanding of user centred design. You should also look at the websites for groups working in this area specifically <http://www.killersites.com/>.

Usability is part of the iterating design and often neglected by technicians who feel that design is a secondary consideration to functionality. Users may not share that perspective and a system the users will not use has fallen at the first hurdle.

### 6.2.3. Security

Security is a thorny issue; it is a cost on the development effort and has very few obvious end user benefits. However, security is a key problem as much of the data in your MLE needs to be kept secure on both moral and legal grounds. Security needs to be included at the design stage. Almost invariably, if there is a breach in security it is due to later adhoc security measure being added rather than considered at the time.

Most projects do not consider security a design issue, however the De Montford University MLE project provided a very secure approach to transactions with key systems to minimise the possibility of a

security breach.

## 6.2.4. Testing

The design itself needs to include both processes for testing the design, and access points to allow testing as the system is being developed. End user testing particularly is vital as the design plate needs to be kept spinning; testing and evaluation are the key processes to keep that happening. User testing is closely linked to the quality of the interface and for many MLEs the quality of the user interface will define the quality of the testing processes.

A good development methodology will have a rigorous testing process included within it. Follow this link to the Project Management infoKit to read more about [User Acceptance Testing \(UAT\)](#)

## 6.2.5. Maintenance

Maintenance is a design issue, it is often thought of as the thing that happens after the system is delivered whereas, in fact, it starts on the first day. Maintainability has to be considered as a design goal, if the system cannot evolve to meet the changing user needs then it will die. Maintenance is about designing in processes to minimise the maintenance costs. You can do this by ensuring that documentation is up to date and that there is an agreed process for checking its correctness, this maybe having someone else come in and check it.

Most JISC projects developed or borrowed documentation practices from their organisations or the methodology they chose.

Other areas where you might have maintenance issues may be in the complex interoperability areas. To maintain the system you need to build a process to track the core suppliers and their products for changes, this about ensuring the 'where are we now?' plate is kept spinning and relevant.

## 6.2.6. Functionality

Finally, there is the functionality of the MLE. So far you will have looked at a number of, what might appear to be, side issues in the MLE, but they are just as important as what the thing does. The user requirements gathering process is the key for functionality criteria, does the MLE satisfy a user's requirements?

You will also gather functionality criteria from all the other process as well. This is a bit unusual as we tend to see the users' needs as paramount. You need to work out what functionality is needed to support all the systems you are integrating, what functionality would be required in a management interface to reflect a change in your institutions organisational structure for example?

At the end of gathering all these criteria you will have to agree a process for sifting, prioritising, removing and reporting all these criteria to ensure that everyone knows what the goal is. Communications with the other spinning plates, and stakeholders (including suppliers) is vital at all stages.

## 6.2.7. Constraints on the design

There are a number of factors that will constrain the options available to you in the design process. A few common ones to consider are: Legacy applications and procurement cycles. The environment of the MLE will almost always contain legacy applications, which constrains the design options in a variety of ways:

- Legacy applications may consolidate a lot of data in a monolithic form, when the desire is to create a component-oriented design

- Legacy applications may only support file-based or screen-based interaction, when you want to use inter-application technologies such as Web Services or RMI
- Legacy applications may enforce workflows which are inefficient or ineffective
- Procurement cycles for some applications can be very long

There are a number of ways to deal with legacy applications:

- Incorporate the legacy application and its constraints into the design, compromising in some areas to accommodate it
- Encapsulate the functions of the legacy system with new adaptor components, such as J2EE servlets, that expose its capabilities using newer technologies. Eventually, the legacy system can be replaced when all dependent applications talk to the adaptors instead of directly to the legacy system
- Plan the replacement of the legacy system within the design

### 6.2.8. External partnerships and agreements

An MLE does not stand in isolation – any institution will have connections (whether automated or not) with a wide range of partners and associations; for example, [UCAS](#), the FE funding councils, [HESA](#), the [NLN](#), and other official bodies. Also, many institutions operate within local consortia, often of both FE and HE institutions.

Because of these external relationships, there will be constraints on the types of information that you need to gather, store, and provide access to. There may also be constraints on the kinds of process reengineering that is possible, in order to maintain harmony with a broader consortium.

### 6.2.9. Legal obligations

There are legal constraints that need to be understood within the design process. The main ones relate to freedom of information, data protection, and accessibility ([SENDA](#)). Follow this link to the [Records Management infoKit](#) to find out more about Fol and Data Protection. [JISC Legal](#) is available to give free advice on legal issues.

### 6.2.10. Capabilities

The technical capabilities of institutions are a factor, as is the degree of outsourcing they are able to manage. Custom software and configurations need to be maintained, so this must be planned for. It may be the case that there will be no capability for supporting custom solutions, so the design is limited to what can be achieved using off-the-shelf products.

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.3. Design Process

Having gathered a set of criteria we need to wrap this all into a process. The process must, on an on-going basis gather and track the criteria and turn them into designs, plans and schedules that evolve with the project and the stakeholders.

The design criteria and its gathering may also be part of the same process. There are many different methodologies for managing a systems design process, and you may have a favourite in mind. If your favourite, has worked in the past, and is well understood by your development team, then it forms a good starting point. If you do not have one, we will suggest some later in design methodologies.

The things that make MLEs much harder to develop and deliver well is the open nature of the process. The MLE bonds a variety of different systems together across an organisation and that is an open

system with constant variation. So we need to pick a methodology and a project management approach that reflects that.

### **6.3.1. Architecture**

A really good paper to read is from the IBM Systems journal, (see the key resources for this section) this gives a very good overview of the information architecture concept and it is relatively easy to see how this can be applied to your MLE project.

The architecture of your MLE is again about ensuring everyone needs to know the right things at the right time and dealing with changing requirements.

### **6.3.2. Interfaces**

A big question for many MLE designers is where does the MLE stop? Your MLE could include a vast range of services and data sources. The JISC INSIDE project based at St. Andrews and Durham identified over 50 possible candidate systems for inclusion in their MLE. The question for you will often be what is 'in' and what is 'out'.

A good technique to use to resolve this is to identify what interfaces each system has. If there are a number of similar systems then, after the first one, you may find that the others integrate very easily. A key outcome from the JISC funded MLE programme is – 'do a small number of things well', rather than trying to be comprehensive and having to reduce the quality.

### **6.3.3. Causal relationships**

In computer science there is much research in to the causal relationship between systems, for example when one system in your MLE does something, like enrol a student, then another set of systems all have to follow the process through. An example from enrolling a student might be:

1. Enrol student (student records)
2. Create debt in the finance system (finance)
3. Allocate course units and timetables (timetabling)
4. Enter the student of the health centre database (Patient records)
5. Print library card (library system)
6. Notify supervisor (enrolments)
7. etc.

This chain of systems is something the MLE will have to interact with, and these relationships need to be understood if you are going to make the system meaningful. Some parts of the process can happen in parallel and some parts are sequential, understanding how your own business processes work is important if the data you deliver is going to be meaningful. The problem of causality is a big issue where human intervention is required, for example where a lecturer has to sign off a course structure before the modules for a student can be added to the various systems. This process may take days and your MLE needs to understand and cope with it.

### **6.3.4. RAD methodologies**

RAD (Rapid Application Development) methodologies are a class of approaches which differ from traditional engineering approaches by having a relatively fast interactive approach to development. If you do not already have an interactive design approach you may want to consider DSDM – Dynamic Systems Design Methodology. This is becoming a common approach in the UK and it has both good books and training courses available. The NCC (National Computer Centre) is championing the use of DSDM and have a number of relevant resources.

No methodology is perfect however and the technical factors and the skill set of the developers will have an impact on the choice of methodology. Different methodologies favour different tools, programming languages and emphasise different parts of the design process.

If you are going to use Java then there is a wealth of material available on the methodologies that can be used. UML (Unified Modelling Language) and OODM (Object Oriented Design Method) are probably the most well known. You should consider them or at least facets of them if you are using Java.

A note of caution is that the transition from procedural languages like Visual Basic and C to object oriented languages like Java or C++ is not trivial for most programmers, and this will need to be borne in mind when selecting a methodology. If the methodology is design for object oriented use you may have to retrain and retool the development team. This can be a positive agenda for change or an unnecessary diversion, but either way it will be expensive and time consuming.

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.4. Design Outputs

The area of risk management needs to be considered before we look at the outputs themselves. The design process, like any other, has risks associated with it and these need to be considered.

Before we examine the design output; the design process is primarily a communications process – you listen to all the points of view and translate those into design documentation and diagrams which you then communicate back to the stakeholders. This cyclic process also impacts on the other spinning plates by modifying the perspectives of all parties in the process.

The key potential failing in any project is a lack of communication. This is one of the hardest things to get right. It does not inspire developers and may be seen as a distraction by others – away from the real work. For your project to be a success you must ensure there is a well defined and clear communications output from every process.

In the design process communications and synthesis are the two key processes. You need to ensure after an iteration of the design, that all parties get updated designs or that updates are available on the project website with emails having been sent out to notify people. You may have to hold meetings to show the design evolution, the level of effort will have to be balanced with all the other layers.

With that in mind let us consider the design process outputs in turn. We stress the process as this is not just the design plans themselves but also the outputs to the other layers. The key outputs from the design process are the designs and the communication of those perspectives. In Zachman's paper it is clear that there is never one design.

The design itself must communicate with a number of different audiences in their languages and the same thing will be described a number of different ways and times.

The major threads of the design process can be split into three broad categories and we will look at each to see the kinds of deliverables you might need, depending on the nature of the project and the skills of the development team.

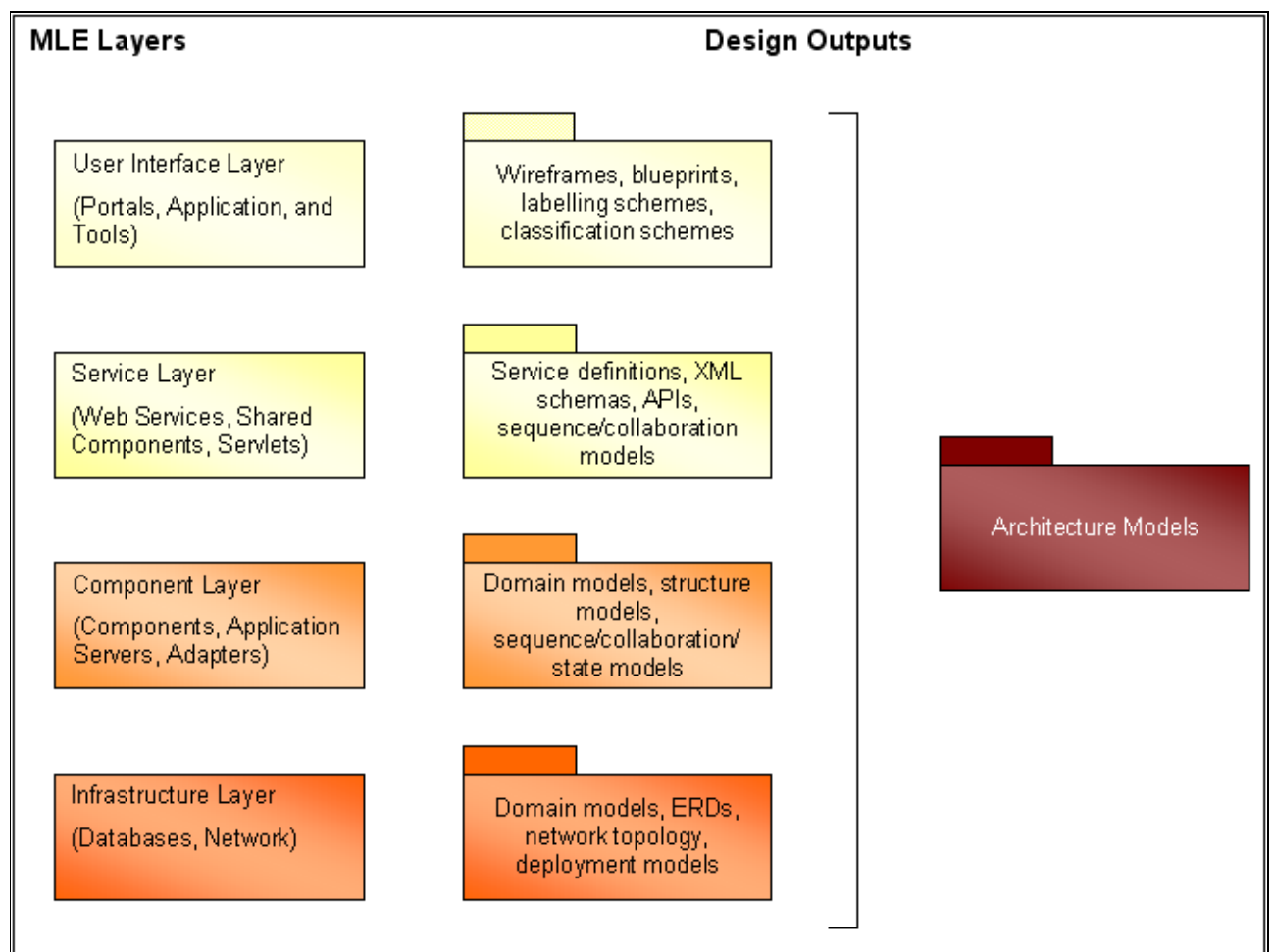
### 6.4.1. Technical

The technical outputs of the design process are those that are going to be used as the goal for software development and integration. These will cover:

- Domain models

- Service interface definitions, data models and XML schemas for web service developers and system integrators
- XML schemas and data models for database developers/administrators
- UML structure, sequence and state models for application and component developers
- Dataflow diagrams for analysts
- Network Architecture diagrams for network administrators
- Systems Architecture diagrams for Operations staff
- User interface designs (wireframes, blueprints, labeling schemes, taxonomies...) for media designers and interface developers

The technical outputs can be related to the layers of MLE technical architecture:



## 6.4.2. Organisational

The design process will move the organisation of both the project and the institution forward. There are two key outputs, the first of which is a soft output concerning the interaction of the project with the wider community which inevitably creates and stimulates change, and this can be very important to long term strategy. The second set of outputs is aimed at the broader management of the institute. The key document here is the overall Information Architecture which provides a schematic of the flow of information and the roles of the major components. This is often used for presentations to senior management. Sarah Holyfield's [paper on diagramming](#) provides very good samples of the kind of diagrams which may help.

## 6.4.3. User

As you design the MLE, you are fundamentally changing both the user expectation and the user's requirements. The first chapter of Alger and Goldstein clearly defines the reality of this aspect of

software development. The user is likely to be interested in the ease of use and functionality and here the user interface designs and the top-level Information Architecture are likely to be the key documents. Since the user is evolving their perspective as they work with the project it is important to iterate designs, prototypes and mock-ups with them as the project progresses.

#### 6.4.4. Conclusion

The process of design is iterative and multifaceted and in it lies the seeds of both triumph and disaster. The design must deliver the functionality required, and inspire and support the project. The design process opens many areas where there is a divergence of opinion and the design process must moderate these while keeping an eye on the prize.

[Follow this link](#) for key resources for this section (these open in a new window)

### 6.5. Design Tools and Techniques

Once a methodology has been chosen – and you should have chosen one – you will then need to look at the tools available to support that methodology. In most cases there will be tools available which vary from simple shareware diagramming and outlining tools, through to very sophisticated development and lifecycle management environments.

In most cases as you investigate and decide on a methodology you will discover the tools which others are using. The tools available usually can be considered in three groups: software development environments, diagramming tools and organisation tools.

Software Development Environments e.g. Rational Rose, Visual Studio and Forte, are designed to support the management and creation of code predominantly:

- Rational Rose is an interesting tool in that it combines a suite of tools to support the whole process from user requirements gathering to software delivery. This tool has a significant learning curve and is entirely directed towards object orientated analysis and design. It is designed to be used with Java and is well known in that community.
- Visual Studio is Microsoft's offering in this arena, and is the host for all the Microsoft development tools and languages. This does not have the extensive design features that tools like Rational Rose has but it is commonly available, and skilled programmers who are familiar with it are much easier to find.
- Forte is Sun's offering for Java development which has a very good development environment but has no design tools to speak of.

Again the interaction between different spinning plates means that decisions cannot be taken in isolation. If you choose a particular methodology, you may be making implicit decisions about the development environment and technologies being chosen.

As the number of possible environments keeps growing, it is important for those in the field to share experience – if YOU have any experience you would like to share please feel free to contribute it to this website through the discussion forum for this section.

#### 6.5.1. Diagramming Tools

Diagramming tools are a key to many areas of the design process. You should read the [JISC paper](#) produced by Sarah Holyfield on this topic for a good grounding. The paper looks at a variety of approaches to diagramming and provides a context for the work undertaken by many of the JISC funded MLE projects. Microsoft's Visio was used successfully by both the [MARTINI](#) project and the [De Montford University MLE](#).

As we have discussed above the design process is highly interactive with a number of different outputs and interested parties. This process is most easily supported by the use of different diagrams and often by using the diagramming techniques with stakeholders. A useful rule of thumb from the INSIDE project is that 'as the audience get higher up the tree the diagrams need to get simpler, ending up with just boxes and a few arrows.'

## 6.5.2. Brain Storming

One area, which is not strictly about the design process, but which is very useful to the designer is the technique of user input through Brainstorming type techniques. This may well happen in the user requirements gathering layer but the output is a great aide memoir and design checklist, and you should be using diagrams to reflect back to the users what is in the MLE.

## 6.5.3. The Unified Modelling Language (UML)

UML, the Unified Modelling Language, is a tool for expressing a whole range of designs. Some of the most useful aspects of UML are:

- Use Case diagrams, which display requirements and users
- Static Structure (class) diagrams, which are used to express the structure of information in the form of classes, operations, and attributes
- Package diagrams, which are used to express dependencies between larger sets of functionality in systems
- Component diagrams, which show how functionality is encapsulated into discrete applications and system components
- Sequence and Collaboration diagrams, which provide a means of modelling communication between systems and services
- State diagrams, which show how a system transitions between states as a result of interaction
- Deployment diagrams, which are used to show how parts of a system are physically configured and connected

Taken together, the UML diagram set is a well-rounded view of any system design, and provides the core information needed by developers to implement a design.

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.6. Domain Models

During any design process it is necessary to produce a number of models that define the outputs of the design process. It is quite common practice to create models which are primarily about the presentation of the design to a wide audience – what is sometimes called 'marketecture', but these aren't the kind of models we're talking about here; what we mean by models in this section are models that do not present a design, they embody the design.

A good set of models are the basis of a design specification, and form the core set of materials that developers and consultants can use to create, deploy and configure systems.

In any design process there will be several models, each of which will look at the design issues from a different perspective. In this section we're primarily concerned with Domain Models.

### 6.6.1. What are domain models?

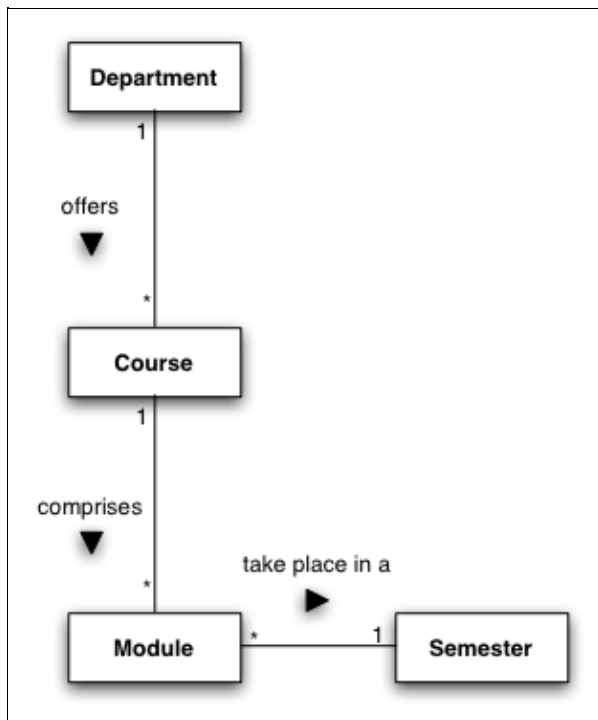
Domain models bridge the gap between the analysis of requirements (x-ref) and the production of design specifications. A domain model represents the common understanding of a key concept in the organisation. For an MLE, these may include the structure of courses, the process of enrolment, or the

structure of assessments. Creating domain models allows the requirements analysis to be embedded into more formal structures that can be the basis of actual implementations.

This is particularly important when we have many different stakeholder groups expressing requirements about the same process or set of objects – we need a common understanding of the structures concerned to prevent inconsistencies creeping into the data models or processes in a 'blind men and the elephant'–type scenario.

Domain models differ depending on whether the target of concern is primarily structural or process–driven, but a recommended technique is to use a fairly basic UML diagram, either a class diagram for structural models, or an activity model (flowchart) for process models.

A rather basic domain model for a course structure in HE might look something like this:



There are several differences between this example and a typical UML class diagram – we haven't included the attributes and operations of these classes for one thing. We've also included some documentation arrows along the associations. The reason for these changes is we want to make the model easier to read for non–specialists – if the model gets positive feedback from the stakeholders and domain experts in the organisation, we can then use it as the basis for creating either UML class diagrams or ERD database models.

(The model above contains some massive flaws that should be fairly obvious, but in case you haven't caught them, one is that it won't support optional models taken from outside of a programme.)

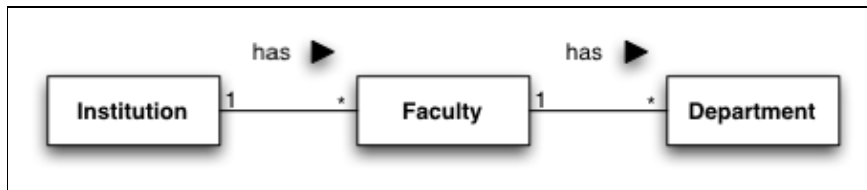
## 6.6.2. Analysis Patterns

Sometimes the domains you wish to model for your organisation share a great deal in common with others, in which case it makes sense to take advantage of existing modelling work. An analysis pattern is a type of model that has broad applicability in many different organisations – it represents a common, well–considered solution to a recurring problem.

The term 'Analysis Pattern' was coined by Martin Fowler in a book of the same name (see Key Resources), to distinguish this kind of pattern from 'Design Patterns', which are reusable patterns at the implementation level. Fowler's book presents patterns for dealing with common structures such as

organisation charts and hierarchies, accountability, currency, time and so on.

When creating domain models, one common pitfall is to create a model that, while fine for representing the current situation, is not adaptable or flexible enough to deal with change. Analysis patterns, which attempt to be generic, usually have more flexibility built into the design. A common mistake is in creating models of organisations which consists of a fixed hierarchy:



This works fine until you change the organisation structure, or merge with an institution that doesn't work in the same way. At this point, if the models you work from are as concrete as the one above, you have a lot of re-development and redesign in store for you. A better approach is to define more flexible models that can accommodate both the present and the set of more likely possibilities for the future. (For a good discussion of models for representing organisation structures, see Fowler's website).

Currently there are very few patterns available for education, but many administrative and financial problems are represented in analysis patterns, so these aspects of the MLE can take advantage of this existing work. The [E-LEN project](#) has been set up to address this lack, and will be creating a range of patterns to cover e-learning and the broader education space.

### 6.6.3. Checking the models

We looked earlier at making domain models readable to non-specialists; well, the main reason for this is that, while domain models synthesize a lot of requirements analysis, the flow shouldn't be one-way. It is essential to go back to the stakeholders and users with any models that are created to check that they really are representing the domain accurately. There are usually subtleties and complexities in any structure or process that don't make it into the first cut of a domain model – partly as a result of implied common knowledge not expressed in requirements gathering, partly out of the desire of modellers to reduce complexity.

### 6.6.4. Developing and refining models

The first abstract domain models are a step towards creating models that can be used as the basis of actual code and data. Your models need to become more detailed and fine-grained with each iteration; you need to add attributes, operations, and more sophisticated constraints and associations. Eventually, it will be possible to take the models as the specification for code, or to even generate the code directly from the models using a Model-Drive Architecture (MDA) approach.

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.7. Architecture

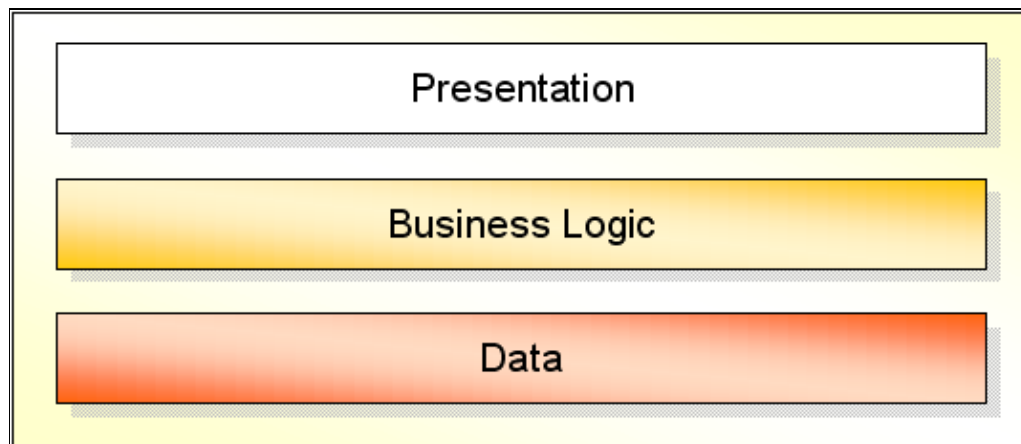
This section builds on the section on [technology options](#) (x-ref), and offers different styles of technical architecture that we can use when constructing an MLE design.

When designing system architecture, it is usually a good start to look at what existing architectural 'styles' are in vogue, and to see if they apply in your case. By choosing a basic approach that is commonly understood in the broader technology industry, it's usually much easier to source products, tools (and personnel) that understand what you are trying to achieve.

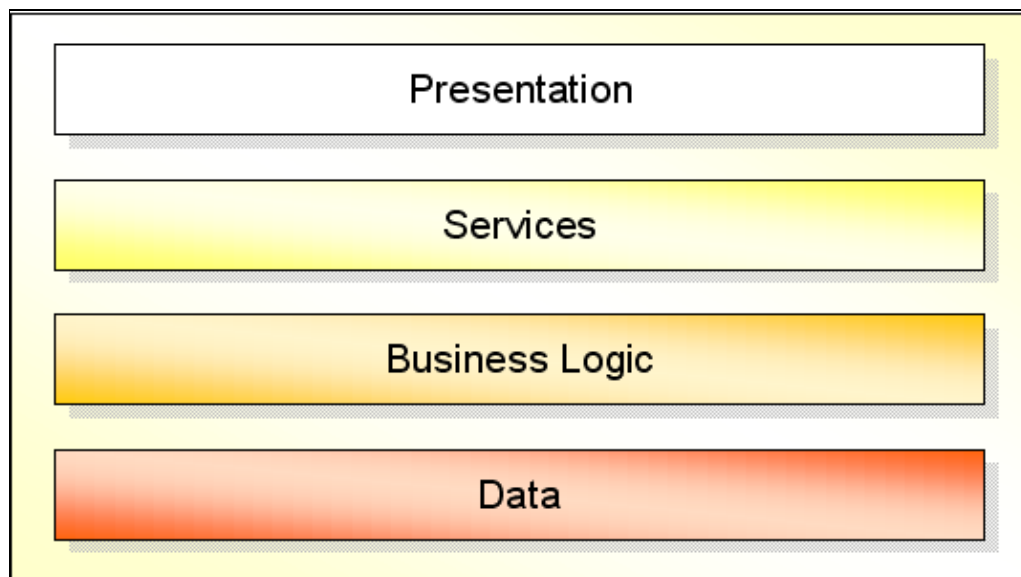
There are five basic styles of architecture with relevance to an MLE, outlined in the sections below.

## 6.7.1. Service–Oriented Architecture

The most pervasive model in the industry today is the service–oriented architecture. In a service–oriented architecture, we take the standard n–tier model of software...



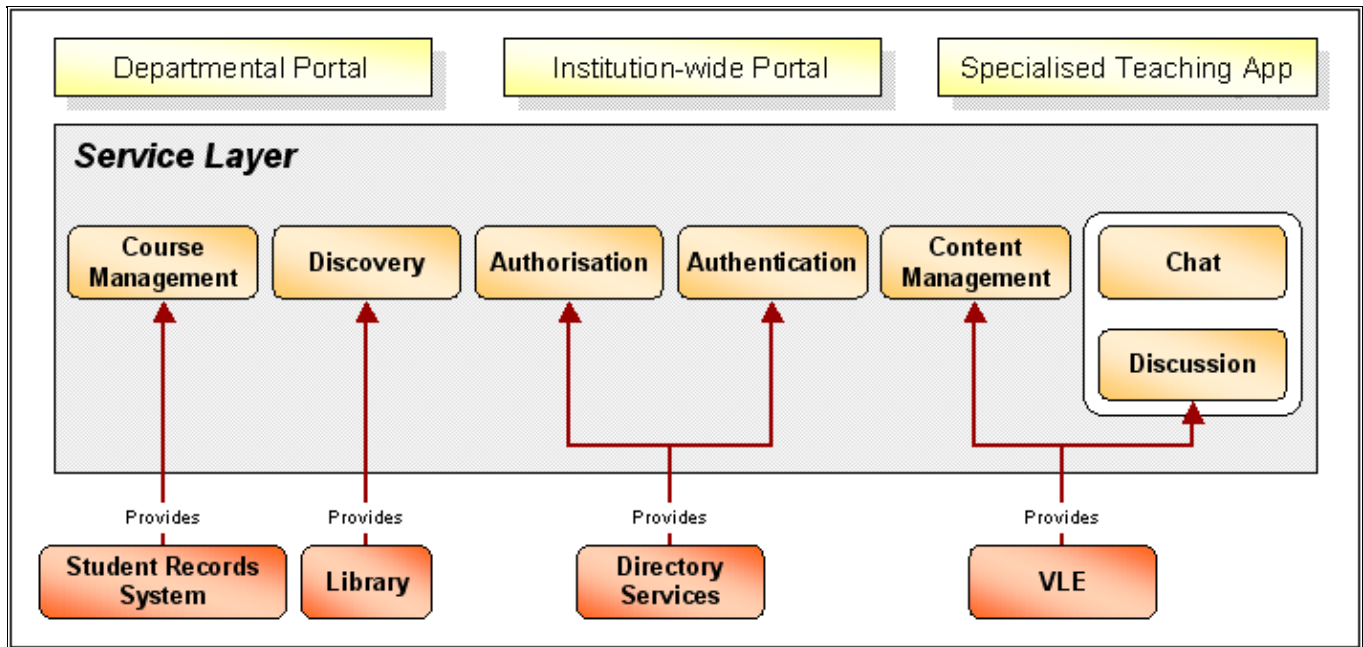
...and add another layer, called the Services Layer, which is interposed between presentation and business logic:



Just as in the n–tier architecture, the business logic layer prevented the direct dependency of presentation logic on the database, so a service layer prevents the direct dependency of the presentation layer on the business logic. Separating presentation from data representation allows changes to be made to the database without affecting presentation; separating presentation from business logic allows the applications that deliver functionality in the architecture to be modified or replaced without affecting the presentation layer.

Or, to put it another way, you can replace or update applications without affecting portals or websites.

In an MLE, the services would tend to be things like Authentication, Authorisation, Course Management, Grading, Content Management and so on. An example service–oriented MLE might look something like this:

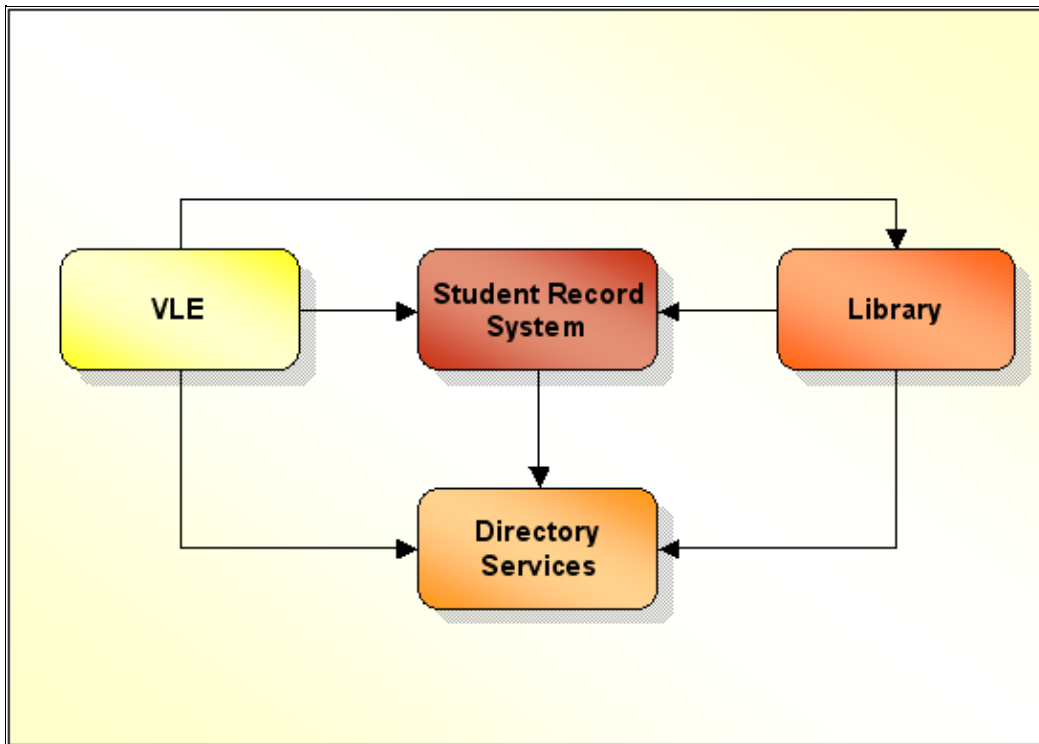


Note that, for the operation of the portals and applications at the top layer, the applications used to provide the service are invisible, and these can be replaced without any recoding or redesign taking place. It is also possible to add on more applications and portals at the top layer without requiring any changes deeper down the stack, which is a problem that affects more traditional EAI approaches (see below).

Service-oriented architectures are almost always based on the use of web services, and this is one of the primary models used in Microsoft's .Net architecture. As a result there are a great many tools, websites and books available to support it. It is also the style of architecture favoured by the Open Knowledge Initiative at MIT and the IMS Global Learning Consortium in its Abstract Framework.

### 6.7.2. Enterprise Application Integration

EAI is a fairly traditional sort of architecture, with its roots in enterprise computing back into the 1980's. In EAI architecture, our primary intent is to integrate the set of existing primary applications to allow sharing of data and capabilities. In an MLE this usually looks something like this:



There are some problems with this approach, however. One particular issue is that it entrenches the division of functions into particular silos in the organisation, and does not address issues such as duplication of services or data. Instead, the approach is to replicate data between the set of systems so that they try to remain in synchronisation.

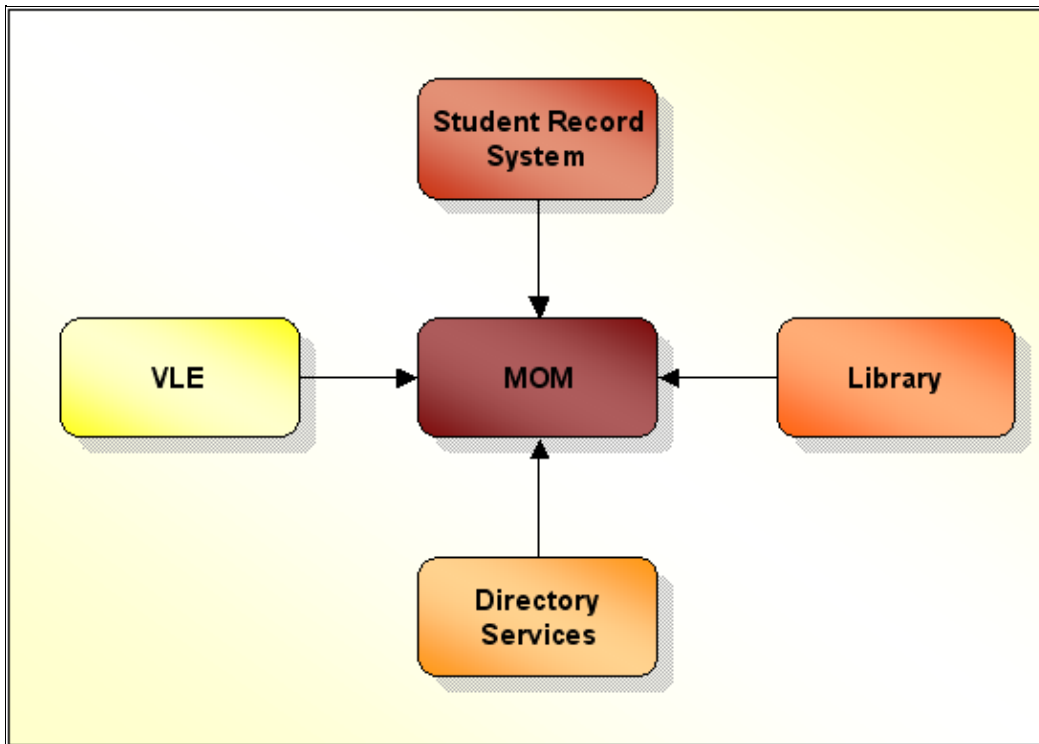
Also, while the problems of integration remain tractable provided the number of systems is very small, as soon as other services need to be integrating – like calendaring, resource management, repositories, finance, HR etc. – then the number of interconnections and replication activities becomes unmanageable,

There are a number of toolkits that assist in EAI; particularly in the areas of data replication and transformation – a number of the JISC MLE projects also used architectures like this one, and developed their own toolsets to assist with integration.

### 6.7.3. Message–Oriented Middleware

A variant of EAI is to use message–oriented middleware (MOM) to provide a framework for handling the data integration issues. A MOM provides routing and transformation capabilities in a central service, reducing the complexity of the interrelations between applications, and enables the MLE to scale. Examples of MOM software include Microsoft BizTalk, Sun ONE Integration Server (Sun's iPlanet Integration Server), and IBM's MQSeries. There is also an open–source MOM toolkit being produced by the SHELL project.

The EAI+MOM model looks something like this:



MOM approaches are a substantial improvement over standard EAI approaches, and may represent a good logical step forward for existing EAI-type MLE implementations that are facing issues of manageability and scalability.

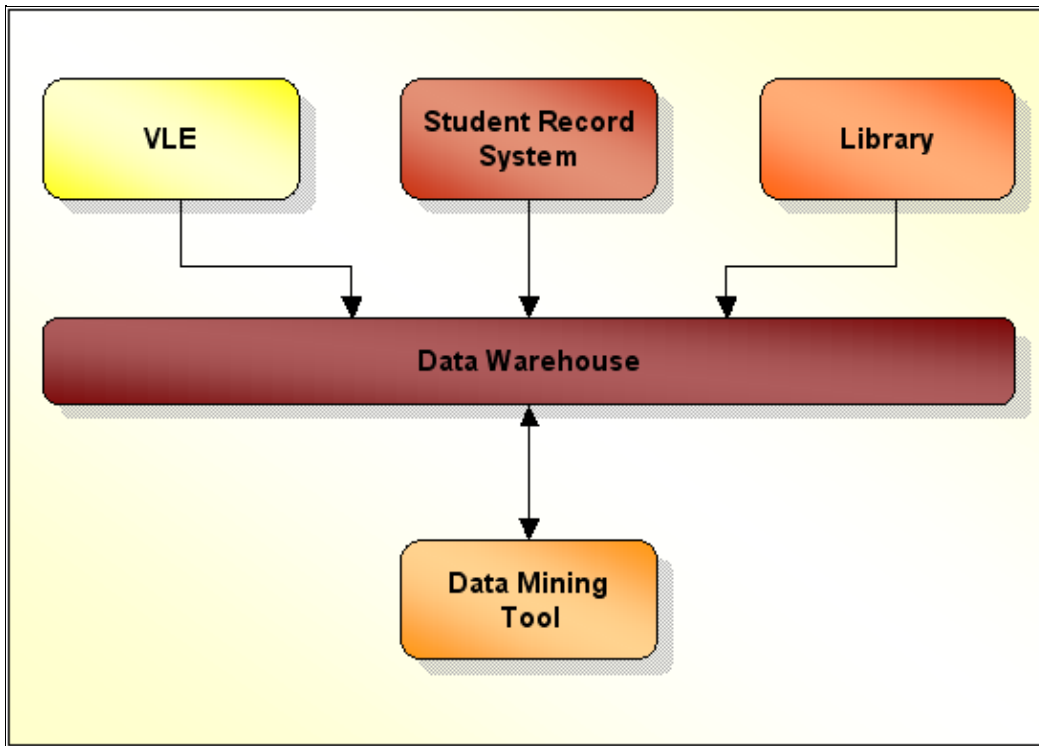
#### 6.7.4. Data Warehouse

A data warehouse model is a very different kind of architecture, which may be chosen to supplement an MLE (or to provide a different sort of MLE entirely).

A Data Warehouse approach involves the integration of data to provide the capability for conducting analyses that support decision making. Data warehouses do not offer any kind of integration at the operational end of the architecture, but instead provide a way to bring together the results of operations for strategic assessment.

A data warehouse relies upon a number of special data modelling techniques (star schemas, also known as multi-dimensional databases or query-optimized databases), high performance data stores, and a great deal of data transformation. Warehouses also require specialized analysis and data-mining tools to conduct assessments of the data and to support strategic decision-making.

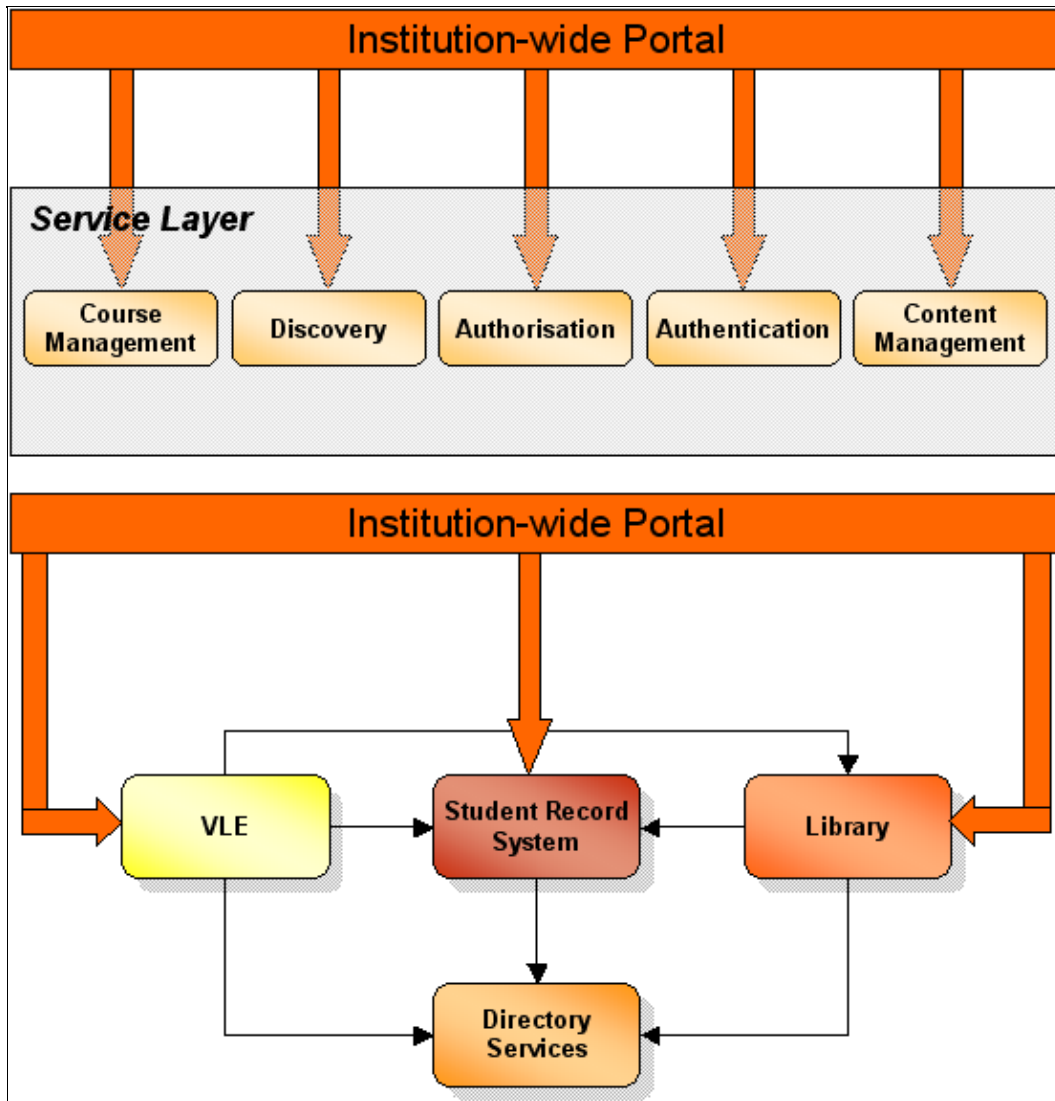
Overall, data warehouse architecture looks something like this:



Data warehouses are extremely costly, and do not have an effect (directly at least) on integration at the operational level. However, large organisations that need to react quickly to changing market environments find them a worthwhile investment; educational institutions on the whole have not investigated their use (MIT being a notable exception).

### 6.7.5. Portal-centric architecture

A common architectural model used in MLEs to date has been to provide a presentation-level integrated view using portal technologies to aggregate various data sources. In practice this tends to resolve itself into an integrated presentation layer on top of either a service-oriented or EAI-type middle layer:



Portal-centric architectures use technologies such as uPortal and associated standards such as the Web Services for Remote Portlets (WSRP) specification to define XML-based 'channels' for each information source.

Portal-centric approaches to architecture need to incorporate a view of how the underlying architecture provides integration, or all that you end up with is a bunch of frames showing inconsistent data and poorly joined up systems on the same web page.

On the plus side, portal technologies generally provide a good framework for thinking about the presentation layer, and integration between either the presentation layer and the services layer, or directly with application logic or (even less advisable) databases.

### 6.7.6. Other approaches

These styles, while representing the vast majority of integrated platforms 'out there', aren't necessarily the only answers possible. Requirements analysis may reveal a set of demands that call for a different approach.

...And finally, the Single Big Vendor System architecture

Any section on MLE architecture wouldn't be complete without a mention of another rather (too) common architectural model – the Single Big Vendor System architecture. In this model, all system parts are sourced from a single vendor as a single integrated application.

This has advantages in the sense that it (usually) works reasonably well at installation, but locks the institution into a single vendor, and usually a single technology platform. (This isn't the same as using a single primary vendor to integrate various systems using EAI or a Service-Oriented Architecture).

[Follow this link](#) for key resources for this section (these open in a new window)

## 6.8. User Interaction Design

This topic has a number of different titles, including Human-Computer Interaction (HCI) and User Centred Design (UCD); it also covers some aspects of information architecture. What we're concerned with generally is the aspect of system design that deals with how human beings work with systems.

### 6.8.1. Analysing tasks

Part of the requirements process for the MLE involves creating use-cases for the various users and stakeholders in a system; Task Analysis builds on this process to create a matrix of users and tasks.

By placing the set of tasks in the rows of a table, and the sets of user groups as columns, we can start quantifying the degree of importance each task has for a particular user group. For example:

| Task                                          | Tutors | Registrars | Students |
|-----------------------------------------------|--------|------------|----------|
| View Student Details                          | 5      | 5          | 8        |
| Change Student Details                        | 0      | 8          | 0        |
| Request Change of Enrolled Module for Student | 1      | 1          | 6        |
| Change Enrolled Module for a Student          | 0      | 10         | 0        |
| View Module Details                           | 8      | 2          | 10       |

In this table, the tasks are weighted by number for different user groups, with 0 meaning 'doesn't do this task' to 10 meaning 'frequently performs this task'.

By creating these weightings, it becomes possible to start grouping tasks together into sets, to become the basis of assessing the navigation and workflow needs of users. Tutors, for example, would prefer to have all the tasks they perform located together so they can be easily navigated.

Task analysis is a useful activity that can inform not only the system design but also the documentation and training strategy for the MLE, as it provides a way for deciding how many different sets of support documents to produce, and how to divide up the material between them.

### 6.8.2. User stories – Personas

In addition to the highly qualitative task-user matrix, another useful technique for informing the interaction design process is a more qualitative approach called 'Personas' or 'User Stories'. This involves making very specific and personal narratives of particular users' tasks – personas are a very vivid means of focussing design on user needs, and help ground the design work. However, if you use this approach make sure you get a fairly representative set or you can easily skew the focus of the design.

### 6.8.3. Workflow

Users don't just perform tasks as standalone units; instead, tasks often have a pattern of flow. This is important in looking at how the MLE needs to operate. If a particular sequence of tasks is a very common activity for a user, you need to consider how to facilitate that workflow within the MLE.

Workflows are often very specific to a particular user group or role, but can be so central to their interaction with systems that significant performance gains can be made by making special provision for it within the MLE.

Other workflows are common across all user groups, and so needs to be incorporated into the overall MLE design (authentication being a common example).

Workflows can be best expressed as process maps using the UML Activity diagram (or 'flowchart' as its also known).

#### **6.8.4. Information architecture**

Information architecture is a very broad subject, but there are some key IA considerations for developing an MLE:

##### *Developing a consistent labelling scheme*

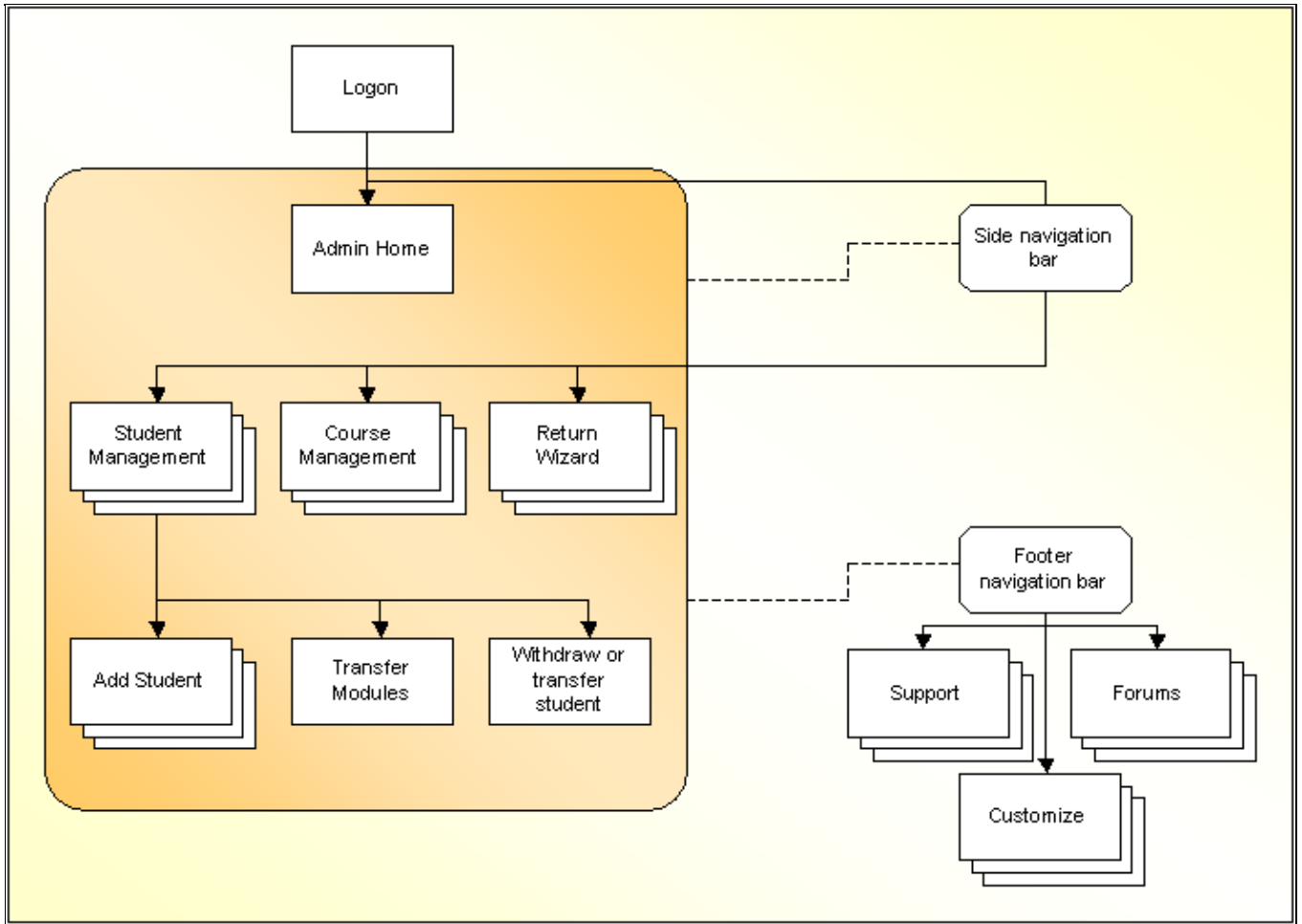
Users are easily put off by inconsistencies in labels used in a user interface. If something is called 'Course Management' in one location, it needs to be called the same thing throughout. Labels must also make sense to users, and bear some relation to the tasks they need to perform.

##### *Developing navigation schemes*

The analysis of tasks and workflows should inform an easy to use navigation structure, tailored to the needs of core sets of users.

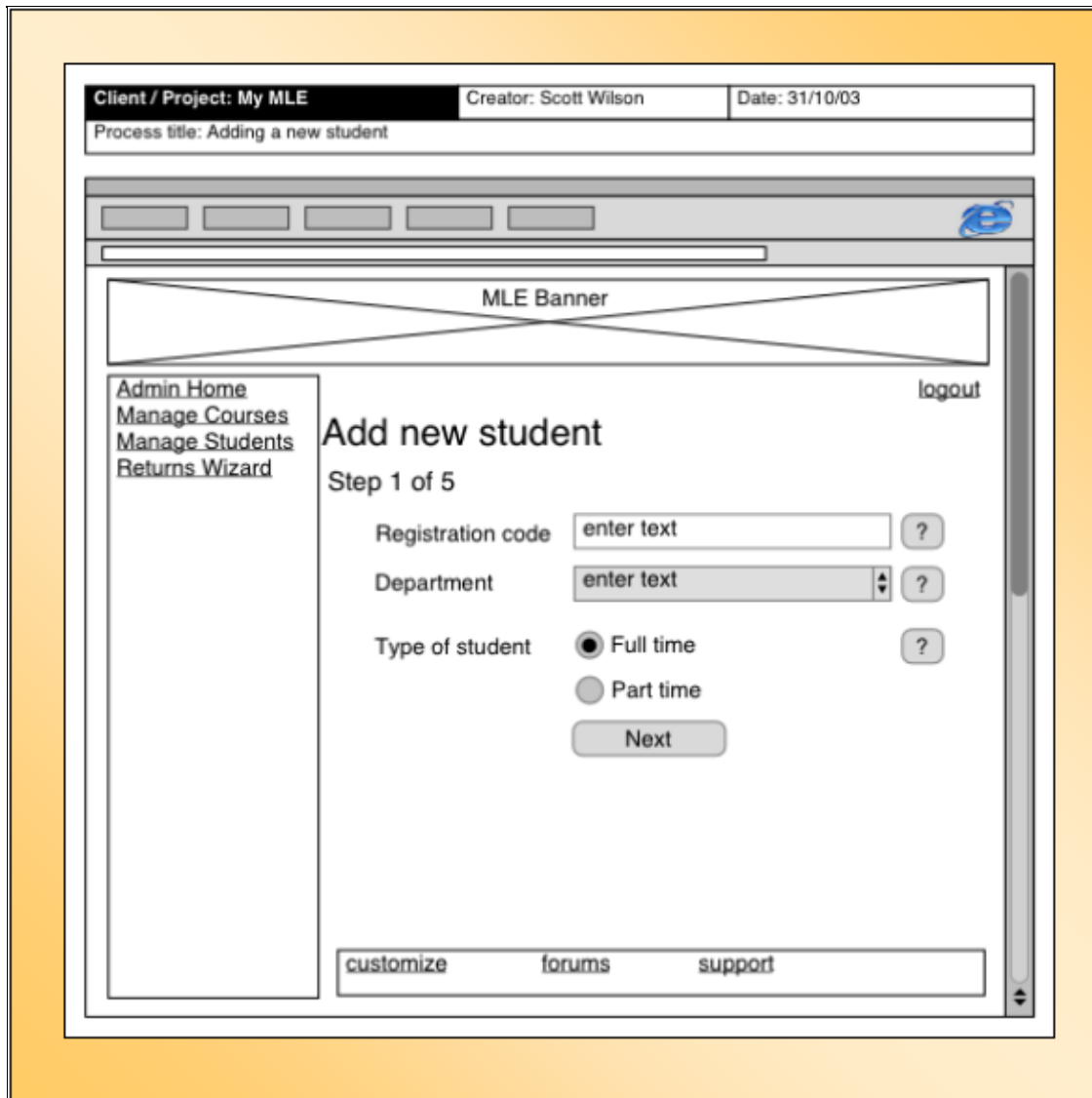
##### *Blueprints for presentation logic*

Blueprints define the structure of the user interface, with the links between web pages or GUI elements clearly defined. A blueprint shows how users can get from one task to another in a proposed user interface. There are a number of different ways of drawing blueprints, and there are some references to these in the resources section. Below is an example of a blueprint for part of an MLE:



*Visual designs*

To assist in the development of the MLE, a set of visual designs, incorporating the navigation and labelling decisions, should be created and critiqued. Visual designs are best worked on using 'low fidelity' models such as wireframes. A wireframe is a very basic approximation of a user interface that allows designers and users to assess whether a design works; here's an example:



### 6.8.5. Evaluating design choices

The interaction elements of a system are the parts of the MLE that staff and students work with day after day, and so they have a lot at stake here. So it is important to carefully evaluate interaction designs with users. This can be accomplished via focus groups and design critique sessions during early design phases, and with usability tests of prototype interfaces with the intended users.

[Follow this link](#) for key resources for this section (these open in a new window)

### Section Editor

**Greg Newton–Ingham** is currently Director of Web, Learning and Network Services at the University of East Anglia responsible for the University's services in these areas. Prior to this role Greg was the founding head of JISCs Advisory service for Moving pictures and Sound and has worked on a number of JISC projects and initiatives.

Greg has had roles, mainly in HE, as consultant, project manager, lecturer and researcher. The key thread of his interest is the place where people and technology interact and the roles that each play in making things work. Greg background is in Computing and Strategic Information Systems where he has been involved in the development of a number of major systems including TLTP, eLib and MLE projects.

## Section Editor

**Scott Wilson** is currently Assistant Director at CETIS, JISC's Centre for Educational Technology Interoperability Standards, which he joined in 2001. He previously worked in the private sector on systems for data warehousing, customer relationship management (CRM), and intelligence analysis (business, military, and policing applications).

He has worked in a wide range of roles in the software industry including solution architect, consultant, QA manager, developer and technical writer.

As part of his role in CETIS, Scott is working with IMS on developing a number of interoperability specifications for web services in education.

---

### **Disclaimer**

We aim to provide accurate and current information on this website. However, we accept no liability for errors or omissions, or for loss or damage arising from using this information.

The statements made and views expressed in publications are those of the authors and do not represent in any way the views of the Service.

The JISC infoNet Service offers general guidance only on issues relevant to the planning and implementation of information systems. Such guidance does not constitute definitive or legal advice and should not be regarded as a substitute therefor. The JISC infoNet Service does not accept any liability for any loss suffered by persons who consult the Service whether or not such loss is suffered directly or indirectly as a result of reliance placed on guidance given by the Service.

The reader is reminded that changes may have taken place since issue, particularly in rapidly changing areas such as internet addressing, and consequently URLs and email addresses should be used with caution. We are not responsible for the content of other websites linked to this site.

This material is licensed under the [Creative Commons License](#) – 2006

---