Application Integration

Management Guide

Strategies and Technologies

Published May 1999, United Kingdom

Butler Group Management Guides

Management Guides provide an in-depth analysis of the business issues and technologies relevant to a particular topic. Butler Group Analysts write a comprehensive overview of the topic being addressed and prominent suppliers are invited to submit their own editorial, including diagrams.

Each supplier has provided its own text for inclusion in this guide, for which Butler Group holds no responsibility for its technical accuracy. For further details, contact Butler Group on +44 (0)1482 586149.

Previous Management Guide titles include:

- Component-Based Development
- Electronic Commerce
- Java Technologies in the Enterprise
- Business Intelligence
- Enterprise Support Management
- Data Webs
- Business on the Web
- A Business Model for Client/Server
- The Business Case for Data Warehousing
- Application Technologies for the Adaptive Enterprise
- Intranet Technologies
- Workflow
- Managing Client/Server Development
- Executive Information Systems

All previously published Management Guides have a dedicated page on Butler Group's Web Site under 'Publications' at:

www.butlergroup.com

ISSN 1356-9384

Researched by:
Lawrence Wilkes

Contents

Butler Group CBD*i* Forum

Butler Group's Forum for Component-Based Development and Integration

The Component-Based Development and Integration (CBD*i*) Forum has established a world-leading position in the area of advanced application delivery. This has been achieved with a small, expert analytical team working with CBD*i* Forum members worldwide, who contribute their experiences in using advanced application delivery practices.

The CBD*i* Forum organises regular meetings in Europe and the USA, and provides a continuous information service on advanced application architectures and practices, including: Application Integration; component-based applications; component modelling and development; assembly and workflow; and componentware.

The CBD*i* Forum focuses on management level information. Members are, typically, middle to senior IT managers responsible for application and/or infrastructure strategy. Members have compared the CBD*i* Forum to an independent user group.

Butler Group provides unique information dissemination and analytical services through the CBD*i* Forum on the application market, the products and services. In addition to conventional industry analysis with input from vendors and customers, the CBD*i* Forum provides access to real-world experience that other industry analysts cannot. With its tight focus on component-based applications and Application Integration, the CBDi Forum is ideally placed to advise IT management on critical application strategies.

Analytical Services:

The monthly INTERACT Journal synthesises the results of CBD*i* Forum meetings and member feedback as patterns, together with market and product analysis. Proceedings, analysis and presentations are also available from CBD*i* Forum meetings, with permanent access via the Internet to all CBD*i* Forum materials in electronic form.

Forum Events:

Forum meetings are attended by members and non members providing the opportunity to create a dialog and network with managers implementing advanced application architectures. Leading vendors also exhibit at the meetings and provide the opportunity to keep up to date.

www.butlerforums.com/cbdindex.htm

For more information contact Phil Storrow
CBD*i* Forum Business Development Manager:

Tel: +44 (0)1244 570955

E-mail: ***phil.storrow@butlergroup.co.uk***

Butler Group

Introduction

This Management Guide focuses on Application Integration, which Butler Group defines as the requirement to integrate into new business processes the functional behaviour, or business rules of disparate systems, or components of them, as well as, but not just, the data that underlies them.

Some developers are puzzled by this sudden focus on Application Integration. They argue that they have not delivered many new 'green field' applications for many years and that integration with existing sources of information has been a common feature for some time.

Application Integration is part of the natural evolution of application delivery that includes improved software componentisation and the increasing acquisition of packaged software.

There is no denying this. Like many things, Application Integration is part of the natural evolution of application delivery that includes improved software componentisation and the increasing acquisition of packaged software. However, the focus in the past has been more on integration of in-house developed applications and components, which is easier when all of their source code is available and controlled within the project or same Information System (IS) department, and can be changed to enable integration. Integration was then just seen as part of the application development process.

Now, Butler Group observes that Application Integration is effectively becoming a discipline in its own right, due to the following trends:

- Greater need for real-time integrity and process integration, not just data exchange and replication.
- New integrated business processes are not only crossing organisational boundaries within a company, but flow between many companies too.
- Constant introduction of new business processes, requiring re-integration of the same core business logic into new applications
- The increasing need to integrate new application workflows with externally developed black-box software, of which implementations are hidden from the developer that can only use their existing interfaces.
- The need to integrate one package with another, where both are black-boxes and the only development task is integration.
- It has become more complex technically. There are more technologies involved, and these are increasingly more complex to use.
- Reduced business change cycle times. Developers cannot respond quickly enough to integration needs by developing all the integration software in-house.
- The productisation of integrationware. The timely arrival of off-the-shelf solutions to common integration scenarios, and the possibility to automate integration tasks.

Thriving on Chaos!

Looking at today's business and technology scenarios, one cannot help being reminded of the following quotes from Tom Peter's book, 'Thriving on Chaos, Handbook for a Management Revolution' *[Thriving on Chaos, Handbook for a Management Revolution by Tom Peters, Macmillian London Limited 1987]* written back in 1987 which, 12 years later, is clearly ringing true:

> "Nothing is predictable"
>
> "We don't know who our competitors will be, or where they will come from"
>
> "We don't know whether merging or de-merging makes more sense, and we have no idea who will be partners with whom tomorrow or next week, let alone next month"
>
> "No firm can take anything for granted …. this is the average scenario for banking, healthcare, public utility, soup maker and computer maker"

Though terms like 'E-Business', 'Disintermediation' or 'Supply Chain Optimisation' might be the fashionable buzzwords today, it really is responding to chaos that Butler Group sees as a key driver for Application Integration. With no time to build new applications from scratch, and no one package providing an ideal or complete match to requirements, then the only timely response seems to be to integrate whatever organisations already have, or can quickly acquire or build.

Organisations need to be cautious in how they respond to this situation. Unplanned, ad hoc integration will only add to the chaos.

Yet, organisations need to be cautious in how they respond to this situation. Unplanned, ad hoc integration will only add to the chaos. Tight integration between today's physical implementations of the monolithic sales order processing system and the monolithic logistics system, might solve an immediate business need, but could create a much larger monolith in the form of a sales order and logistics processing system, that is now even more inflexible to change.

Butler Group believes that Application Integration is a permanent state that requires architectural foundations, which enable continuous, efficient and rapid reaction to seemingly random events. The purpose of this Management Guide is to provide a short, but comprehensive introduction to this critical subject, and to provide frameworks for managers to communicate the critical issues that must be addressed to establish a reactive environment.

The Challenge of Application Integration

Though the need for data consolidation and synchronisation has been apparent for some time, and fuelled the growth in technologies like Data Warehousing, the need to just replicate or query and analyse data is giving way to a requirement for real-time availability and, more importantly, integrity of data. For example, stock levels available to e-commerce sales systems need to be precise at the moment a customer decides to place an order, and competitive advantage will go to those organisations who can process that order instantaneously, rather than wait for overnight replication of data. Supply Chain Optimisation and Call Centres are two examples of business activities that need to bring together and process information in real-time to be most effective.

What Butler Group observes as the current drivers for integration are illustrated in Table 1, which, though the priority will vary by industry, Butler Group expect most readers will recognise. Of course, there is no reason why many of these business drivers could be equally supported by the delivery of wholly new systems. However, the importance of time-to-market and resultant competitive advantage, as well as competitive response, means that organisations do not have the luxury of being able to simply replace systems to meet these new requirements. Even if they could, the trend of mergers and acquisitions shows no sign of abating, forcing the continual need to bring different organisations information systems together. In addition, the outsourcing of parts of the business process to specialist organisations continues, but now, because of the real-time information needs, it requires the information systems of the service provider to be tightly integrated as well.

For some of these business drivers, integration is, in fact, a key part of the solution. By their very nature, virtual organisations and supply chains require that each participant integrates their information services with the others, as they all become part of an extended network. The term 'disintermediation' is used to describe the removal of middlemen from the supply chain, often claimed as a key benefit of e-commerce. In reality it is more a process of re-intermediation, as one set of middlemen are swapped for another, for example, removing retail stores as a sales outlet only to have to put a home delivery service in place to get goods to the customers. Butler Group believes that the need to constantly re-evaluate the effectiveness of the supply chain will bring a constant need for re-integration of applications.

Business Drivers	Technology and IS Challenges
• Supply Chain Optimisation. • E-Commerce and Web Storefronts. • Customer Relationship Management. • Call Centres and Customer Care. • Customer and Employee Self-Service. • Mass Customisation. • One-to-One Marketing. • Business Process Improvement and reduced business change cycle-time. • Acquisitions and Mergers. • The Virtual Organisation and outsourcing of services. • Disintermediation and re-intermediation. • Real-time access to information and integrity of data.	• Packages are not a complete solution and inflexible to change. • Need to integrate legacy. Too costly to replace and no time to rebuild. • Need to include new function. Areas of competitive advantage not available in packages yet. • Lack of single standards for technology or business semantics. • Lack of interfaces. • Complex Technology with heterogeneous platforms, *n*-tier distributed computing and the Web. • Incompatible business semantics. • Skills and resources shortages. • Need to balance tactical and long-term solutions.

Table 1. Business, Technology and IS Challenges

Unfortunately, few, if any, of the parts of the jigsaw will have been designed or built with future integration requirements in mind.

As such, the requirement to integrate applications and processes, as well as data, has become the new high-priority for many organisations. Unfortunately, few, if any, of the parts of the jigsaw will have been designed or built with future integration requirements in mind. Only recently, have some package vendors begun to deliver interfaces to their applications to facilitate integration, or operating system and middleware solutions become more interoperable. Even so, they often still require the adoption of an architecture, (with them, of course, at the centre) which is incompatible with the architecture of other major parts that an organisation needs to integrate.

It is the lack of common business and technical architectures that makes integration complex. It is not just the differences in data formats or interfaces, but the lack of common definition of the business concepts that underlies the different sources to be integrated. Mapping a 6-digit ASCII number to a 10-digit EBCDIC field is perhaps straightforward. However, mapping, for example, customer data that contains the concept of multiple locations in one source, into another that does not, is clearly more complex, and having to update sales order data simultaneously into several sources that have incompatible semantic definitions of what exactly a sales order is, steadily becomes a nightmare. Yet, this is not an untypical situation.

IS departments, therefore, struggle to bring incompatible resources in the form of packages, platforms, and legacy systems together. This is compounded by the shortage of appropriate skills, not only for the existing applications, but also for the new Web and distributed computing technologies that new applications must be delivered in. It also has to balance the immediate time-to-market needs of the business with a long-term solution that avoids this scenario being repeated over and over again.

Application and Integration Evolution

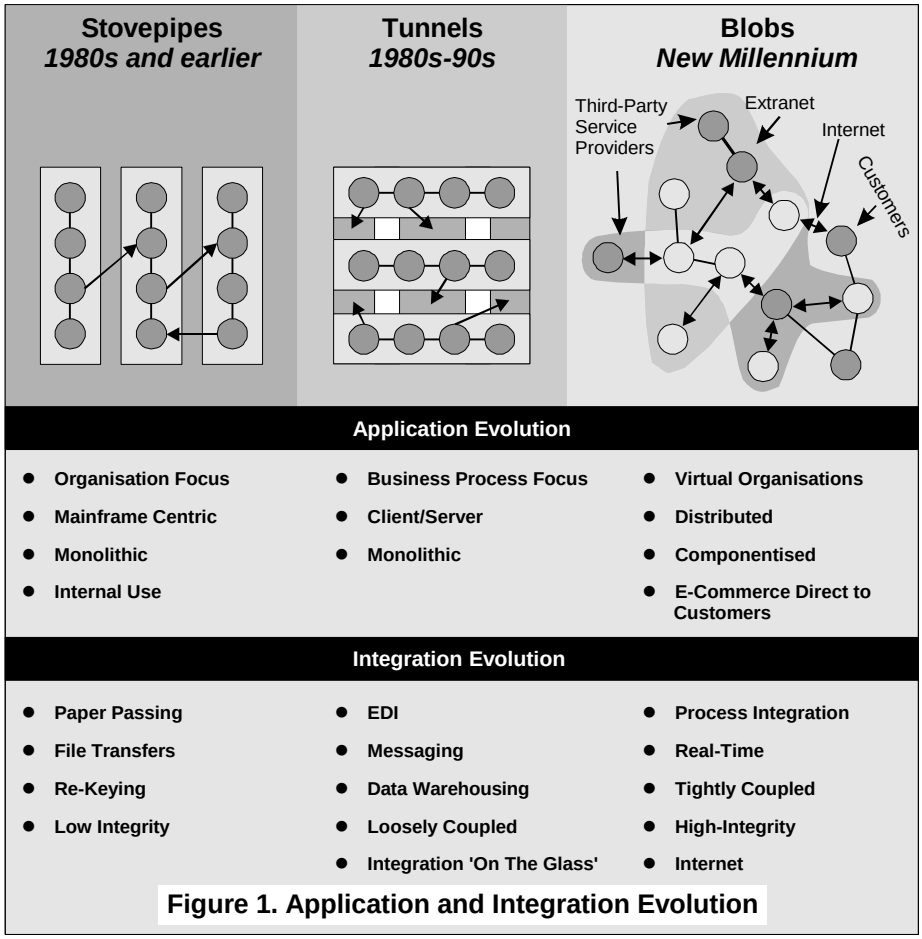

Figure 1. Application and Integration Evolution

Butler Group believes that many of the requirements for integration can be summed up in Figure 1. In the Stovepipe and Tunnel era's, whilst integration within applications might have been tight, integration between applications was much looser and, consequently, the real-time integrity of data was often weak. In the new millennium, applications will become much more like amorphous blobs that are much more difficult to manage, as many of their parts might be beyond your control and in the hands of your suppliers, partners and customers, so integration through interfaces is the only way to bring an application together. Yet, at the same time, the demand for real-time integrity is stronger. It is these factors that are focusing the attention on integration.

Enabling New Business Processes

A common view of the integration requirements for new business processes is shown in Figure 2. There is a requirement for both new front-end process that face the customer, and for back-end processes that integrate partners and suppliers, either of which can flow across (*intra*) or between (*inter*) organisations.

Common Integration Challenges

This scenario, in Figure 2, highlights some of the following common challenges in Application Integration.

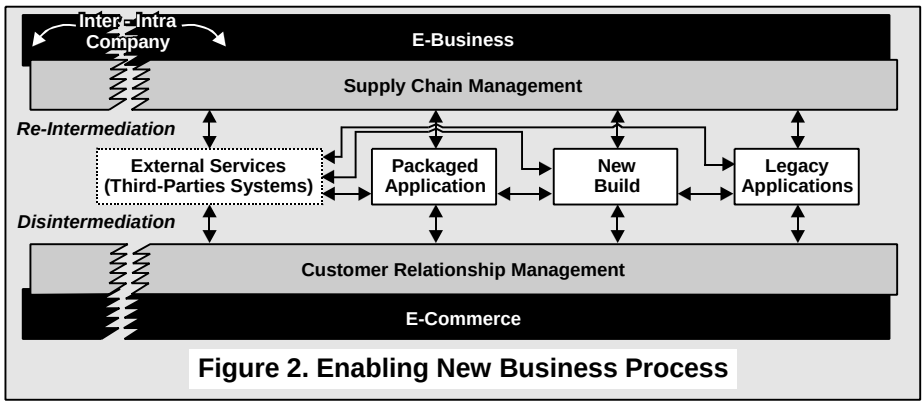

Figure 2. Enabling New Business Process

Consolidation of Multiple Sources

Organisations frequently have multiple systems implementing a similar business concept. For example, few organisations have a single system or shared component in which customer information is held. A frequent integration task is to consolidate these sources into a single view. Whilst it might seem sensible to replace them with a single shared component, the effort to do so can be considerable, especially in the case of legacy or packaged software. Even if it were accomplished, the merger and acquisition trend would probably see yet more sources of the same information arriving shortly after! As a result, enabling the integration of multiple sources can be more attractive than trying to replace them with a single source.

Assembly from Disparate Sources

Developers, that are implementing new applications, face a bewildering array of programming languages, interfaces, protocols, and technologies that must be integrated.

Organisations rarely have a single approach to implementing systems. Sources to be integrated are usually developed and executed in different technologies, with some having been built in-house, whilst others acquired. Developers, that are implementing new applications, face a bewildering array of programming languages, interfaces, protocols, and technologies that must be integrated. Butler Group can categorise these sources as:

- **Common Business Infrastructure or Back-Office Packages** – Acquired packages that automate the common business infrastructure, for example, Enterprise Resource Planning (ERP), Human Resources and Financial Accounting systems that are core to most organisations. (For example, SAP R/3, Baan, JD Edwards, and Oracle Financials, etc.)
- **Front-Office Packages –** Acquired packages that are used to implement front-end business processes, for example, Customer Relationship Management (CRM), sales order processing, and sales force automation, increasingly with an e-commerce requirement (for example, Siebel, Vantive, and Clarify).
- **Legacy Applications** – The portfolio of existing applications built in-house.
- **New Build** – New applications or components built in-house.
- **Third-Party Services –** Information sources or processes that are held and/or performed by third-parties, for example, credit authorisation. Increasingly, this will also include straightforward core business processes, such as a third-party ERP or sales order process, that form part of your virtual enterprise supply chain.

Real-Time Integration

As stated, Butler Group sees increasing requirement to deliver Real-Time Integration (RTI). Moving data between applications using file transfers or replication is inappropriate to ensuring that data is accurate in real-time across all systems, which businesses will require.

Constant Re-Integration

Integration is seldom a one-time effort. Unfortunately, many organisations treat it as such.

Integration is seldom a one-time effort. Unfortunately, many organisations treat it as such. Applications to support new business processes typically require the same core set of systems to be accessed and integrated. For example, a new branch-office system, a mobile sales automation, an e-commerce store front, and a telesales operation will all require access to the same customer, logistics and finance information. Yet frequently, the integration effort that serves the first of these new business processes is often not reusable by the subsequent, for example, embedding integration rules into a client layer that is inapplicable to other workflows.

Legacy Icebergs

Often, only part of the functionality of an existing system needs to be included in a new application. Unfortunately, the dependencies this has with other parts of the existing system and other systems means it is not easy to extract and reuse, and must, therefore, be left *in situ*. This is the iceberg scenario, where the area of interest appears small and might be easy to replace with a new component that is more straightforward to integrate but, in fact, cannot be replaced without rebuilding all those dependencies.

Fixed and Rigid Structures

Applications requiring integration usually have fixed structures for their interfaces, messages or databases. This rigid structure has to be known to the application that wants to integrate with them, and any changes to the structure requires change to the application that are integrated with them.

Integration is a Multi-Step Process

New integration requirements are rarely as simple as a wire between applications ‘A’ and ‘B’ that enables one to send data to the other. RTI requirements of new business processes give rise to complex, multi-step negotiations between multiple applications before a process can complete and ensure that all the data is correct. Integration and the new business process appear entwined, and some vendors sell integrationware specific to this point. However, organisations need to be cautious that they are not bound together so tightly that the process, the sources being integrated, or the integration technologies, cannot change independently, giving rise to an even bigger monolith.

Integration is Application Development

Despite claims by some vendors that Application Integration can be achieved in some automated declarative way, simply plugging-in off-the-shelf connections between off-the-shelf applications, Butler Group observes that significant amounts of manual effort is still required in many cases. Whilst some of the drudgery of transformation might be automated, and there are clearly some common packages to package connections repeated across many organisations and, hence, worthy of productising, there will be high-instances of integrating legacy applications, and expressing complex rules that require what is essentially application development. Application Integration is not an alternative to application development but, as we have expressed, is significant enough to become a specialised discipline of it, in the same way we have database administrators, SAP specialists, or corporate architects, etc.

Lack of Common Architectures and Standards

The root cause of the problems that arise in Application Integration is the lack of common architectures. Whilst it is usual to apply a single architecture and set of standards to an application, it is less common to find different applications conforming to the same set, even within the same company.

This, in part, just reflects the natural evolution of architectures, standards, and technologies and that no one is likely to go back and reapply them to applications already delivered. Also, each acquired application has a different set and is even less likely to be compatible with in-house applications. The consequence is that, in most situations, it is not that there are no architectures or standards, rather that there are multiple occurrences of them.

It is unfortunate that perceptions of integration often seem to revolve just around the technology layer.

It is important to recognise that architectures and standards apply at many levels, as illustrated in Figure 3. It is unfortunate that perceptions of integration often seem to revolve just around the technology layer. In reality, aligning business architectures and their implementation in applications is much more complex, and is where most of the effort in Application Integration will be spent. Insisting all applications have Microsoft COM interfaces or use IBM's MQSeries might remove the technology barriers to Application Integration, but as a simple example, do nothing on their own to reconcile the fact that one application considers location information as an attribute of customer, whereas another maintains it as a separate entity. Application Integration requires communication at all of these levels.

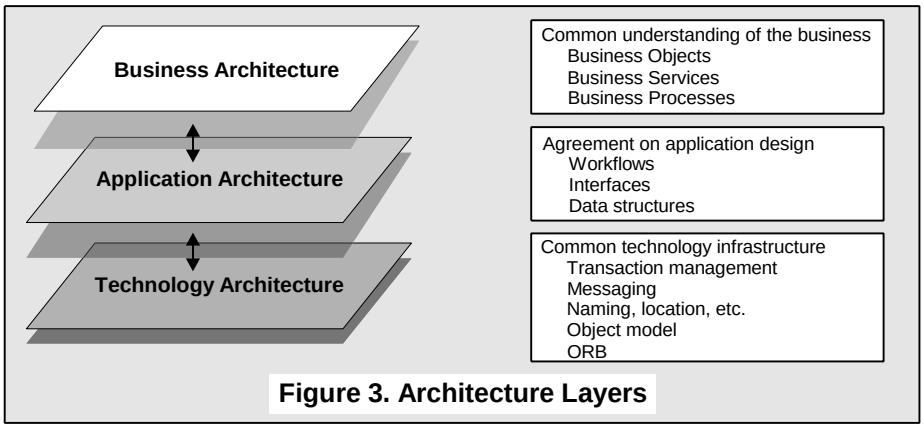

Figure 3. Architecture Layers

Integration Complexity

Attention to each of the architectural layers, combined with the focus on real-time integration, reveals Application Integration to be more complex than first thought, and a reason why organisations need to concentrate on it, in its own right, more so than they did in the past. The following shortlist highlights some of the considerations that must be made in integration tasks, which are also discussed in the next section:

- What are the differences in interfaces, protocols?
- Are there differences in the platform technologies used?
- How are messages and data formatted? Are the data structures similar?
- What event triggers the action, when and how is it invoked?
- How are messages delivered? What guarantees of delivery are required?
- Is real-time transaction integrity required? Is two-phase commit needed?
- Is object behaviour required? Do all the sources need to be wrapped as objects?
- Is it to be Web-enabled?
- Are there business rules governing the integration? Is the integration a multi-step process?
- Are the business concepts the same?

Integrationware – Converging Market

As a consequence of all the above, vendors have been quick to identify an opportunity to introduce a new category of software that Butler Group label 'integrationware'. Some promise to simplify the Application Integration process to the extent that it is almost automatic, or require as little development effort to implement as the off-the-shelf application packages that they can integrate. The following list highlights the diversity of products that vendors are attaching the 'integration' label to:

- Enterprise Application Integration (EAI).
- Application Servers.
- Development Tools.
- Wrapping Tools.
- Workflow.
- Middleware and Messaging Products.
- Packaged Applications.
- Database Gateways.

With so many vendors offering integrationware, there is a bewildering choice. Some are appropriate for tactical point-to-point solutions, whilst at the other end of the spectrum, others may provide a long-term architecture for continual integration needs. Some are general purpose, whilst others are specific to a particular integration scenario. This is examined in more detail in the 'integrationware Market' section of this document.

What's the Problem with Application Integration?

Whilst Application Integration may solve the immediate requirement, Butler Group are concerned that simply layering more spaghetti on the spaghetti will render subsequent integration tasks (which *will* occur) more complex.

Whilst Application Integration may solve the immediate requirement, Butler Group is concerned that simply layering more spaghetti on the spaghetti will render subsequent integration tasks (which *will* occur) more complex, and make the legacy problem even more acute. Areas for concern with Application Integration include the following:

- Integration might solve immediate problem, but what about the next generation of requirements?
- Additional layers of integrationware might impair performance, particularly where interpretation has to be done in real-time.
- You need to balance tactical versus strategic approaches.
- There is a confusion of styles, tools, approaches, and technologies for Application Integration. It is not always obvious which one is right, and clearly there is no 'one size fits all'.
- It should be recognised that it can be more complex to develop and implement the integration than the parts it connects.
- Testing applications and locating the source of faults in the system will be more complex.

It's an Issue That Won't Go Away

Even so, Butler Group does not see Application Integration as a short-term trend. Whilst it might seem to be the most expedient way to meet an immediate business need, Butler Group does not see it as some disposable activity that is left behind once some new applications with a better business fit are delivered to support the original need.

Firstly, history teaches us that these applications frequently do not get delivered, and the legacy systems need to be maintained (in this new scenario re-integrated time and again) far beyond their original life expectancy. However, more importantly, Butler Group believes that, even if new applications were delivered, within a short space of time their business fit would no longer be perfect, requiring yet another cycle of integration to support new requirements, and so the cycle continues.

Butler Group does not believe this should be seen as a problem, rather that through better componentisation of applications, and improved integration layers, it is actually desirable to be able to constantly re-assemble applications in response to changing needs. Butler Group expects larger enterprises will establish architectures and infrastructures that make rapid integration business as usual.

Increasing The Scope of Application Integration

Given this, Butler Group believes that it is important for organisations to take a broader view of Application Integration than their immediate needs might dictate, by considering some of the following:

- Do not just have an internal focus, and consider how integration with external systems might be rapidly achieved if required.
- How easily could a business process be re-integrated if an outsourced service provider replaced one of the internal information sources tomorrow? Would it need re-integration at all?
- Are there standards that could be applied at each of the architecture layers providing compatibility with others in your industry to facilitate integration, for example, in virtual supply chains?
- Could a new business process or workflow be added that reused existing components and systems integrated into an existing application without having to redevelop all the integration software?
- Is provision being made to enable more sources of the same type of information to be easily integrated as a result of mergers and acquisitions?
- Are sources of information that need to be integrated being encapsulated behind interfaces so that their physical implementation can change without requiring re-integration?

What is Application Integration?

Butler Group defines Application Integration as the requirement to integrate into new business processes the functional behaviour or business rules of disparate systems or components of them as well as, but not just, the data that underlies them. Application Integration involves:

- The transportation and transformation of information between one or more applications.
- The timing and sequencing rules that govern when the transportation and transformation takes place.
- The integrity constraints that determine the success or failure of the integration.

Even with this definition, it is clear that Application Integration can be viewed as a broad topic. Users have diverse integration needs that are illustrated in Figure 4. All of these scenarios exist in most organisations. The data integration and synchronisation scenarios are well established in most organisations, but are typically post processes, attempting to ensure the integrity of data after an event has happened, as in common overnight replications of data from one system to another.

However, as illustrated earlier in Figure 1, integration requirements are evolving and Butler Group now sees a greater requirement now for those on the left of Figure 4. A key difference is that the functional behaviour of the systems is required to be integrated, not just the data. In addition, new business processes demand that this must take place in real-time. A further difference is that the integration is two-way, with information being created and updated on both sides of the integration, not just read.

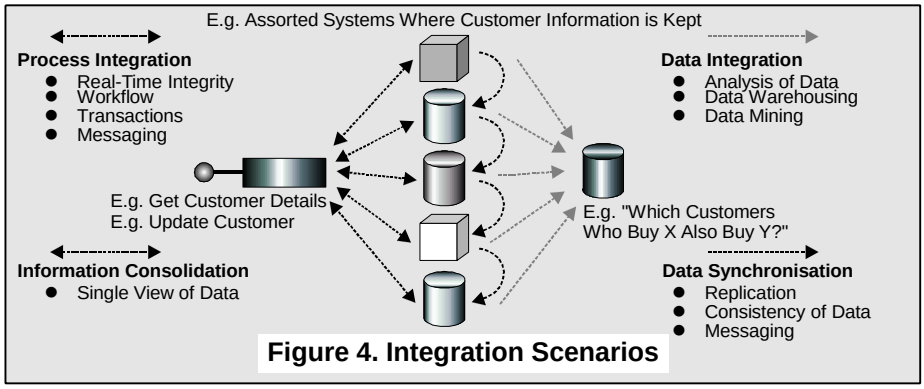

Figure 4. Integration Scenarios

Integration Mechanisms

To support these scenarios, a variety of mechanisms are available to enable integration (as illustrated in Figure 5) which Butler Group can separate into the following three categories:

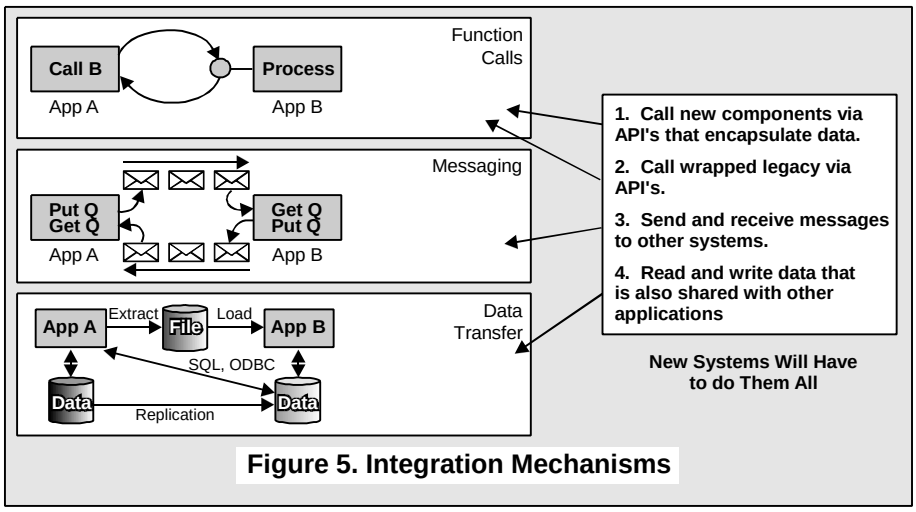

Figure 5. Integration Mechanisms

Call Interface

Applications provide a callable interface, usually referred to as an Application Programmable Interface (API). This does not have to make use of object technologies, such as COM and CORBA, though increasingly they will be, as they provide wider levels of interoperability than more proprietary interface standards. Examples include:

- Object interfaces such as COM, CORBA, or JavaBeans.
- Transaction processing interfaces such as IBM Customer Information Control System (CICS), or BEA Tuxedo.
- Packaged application interfaces such as SAPs BAPI (Business API), which, themselves, can made available in object interfaces such as COM.

Advantages

- Need to ensure the real-time integrity of transactions and data.
- Encapsulating the implementation behind a common interface.
- Need to invoke business logic, not just retrieve data.
- Building new components.
- Providing wrappers around existing systems, which can often be achieved without changing the source of the system.

Disadvantages

- Might be complex to program, and new technologies require the use of object approaches.
- There are many different technologies, though reducing this factor is one of the major attractions of integrationware.
- Synchronous behaviour requires the applications and the connection between them to be up and running.

Messaging

Applications are integrated by send and receive messages, usually via some queuing mechanism. Examples are:

- Message queuing products, such as IBM MQSeries or Microsoft MSMQ, or message queuing interfaces to subsystems, such as IBM CICS transient data queues.
- Application package messaging, such as SAP's Application Linking and Embedding (ALE).
- Mail systems and groupware products, such as Microsoft Exchange or Lotus Notes.

Advantages

- Enables asynchronous, loose coupling of distributed applications.
- Implementing integration between organisations, where availability of applications cannot be guaranteed.
- Can be used for a 'Publish and Subscribe' approach, where the sending application requires no knowledge of what applications subscribe to its messages.

Disadvantages

- Requires applications to use the messaging interface, and know when/how to read and write the queues which, therefore, requires the code of legacy applications to be changed.
- Can require extra effort to add synchronous, real-time behaviour on top of messaging systems.

Data Access/File Transfer

Applications are integrated by direct access to their databases, or via file transfers. Examples include:

- File Transfers and batch loads.
- Direct read and write databases using database calls and database gateways, such as ODBC or Information Builders EDA/SQL.
- Replication provided by database management systems.

Advantages

- Useful when there are large volumes of data to move.
- Supports off-line analysis and reporting on large volumes of data.
- Can be straightforward and easy to implement.
- Does not require the existing application being integrated to be changed.

Disadvantages

- Low integrity, as replicated data is out-of-date.
- Low integrity, if business rules and validation of existing application are bypassed.
- Does not encapsulate physical implementation, and new applications are affected by change to the ones integrated.
- Does not integrate business rules of the existing application.
- Data may require interpretation to be turned into information.

Whilst these are three distinct mechanisms, typical integration projects to support new business processes will frequently use a mixture of them all (as also illustrated in Figure 5) as this is governed by:

- The availability and suitability of the interfaces offered by the existing applications being integrated.
- Each integration connection in the new business system might require a different behaviour, for example, some synchronous, others asynchronous.

Conversion and Transformation

One of the key challenges of Application Integration is provide connections between incompatible instances of the above mechanisms. Additionally, there is often a need to mix these mechanisms, particularly in the case of integrating black-box packages and components or legacy systems, where neither end of the connection can be readily changed to suit the other. Typical conversion tasks occur due to incompatible:

- **Interface Technologies –** For example, conversion of IBM CICS to Microsoft COM, or COM to CORBA bridges.
- **Transport Technologies** – Incompatible messaging products and network protocols.
- **Data Types and Structures** – Conversion of data-types is a common activity. More complex is the formatting and restructuring of incompatible data structures in interfaces or databases.
- **Integration Mechanisms** – For example, application ‘A’ might output messages to a queue, but application ‘B’ is only accessible via an API.

Integration is a Business Process

As mentioned in the new business scenarios, Application Integration is rarely a single step. For example, fully integrating a new e-commerce sales system may require access to customer, product, logistics and finance systems, all in real-time.

The workflow of a new business process provides the context for integration. Each stage of the process might, itself, involve several steps of integration crossing disparate sources of information in order to complete, as illustrated in Figure 6.

Integration is not just a simple transfer of data, but also contains rules that govern the sequencing and ensure the integrity of the transaction.

As such, integration is not just a simple transfer of data, but also contains rules that govern the sequencing and ensure the integrity of the transaction. Each step may have to complete before the next can commence. However, it is not obvious where these rules should be coded. Are they part of the new application, part of the existing applications, or should they be in a separate integration layer?

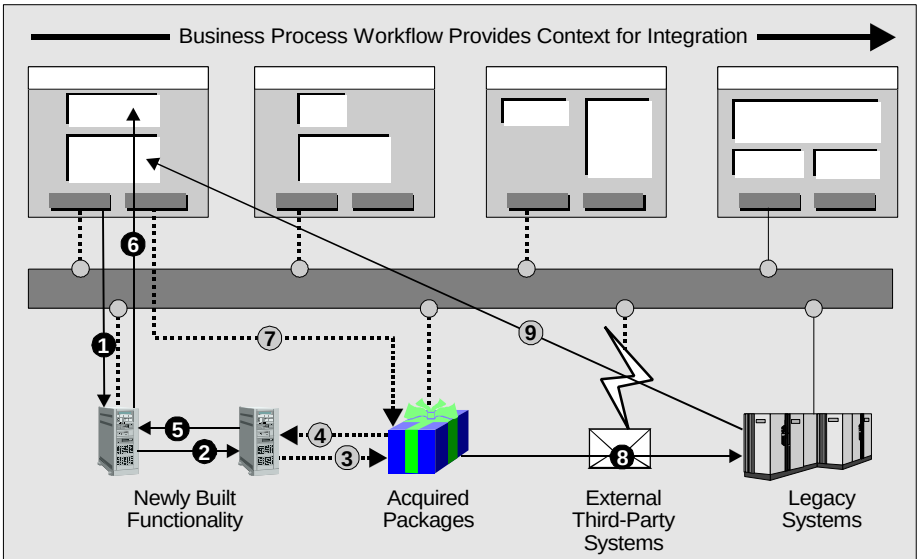

Figure 6. Integration is a Process

Avoiding the Perpetuation of Legacy Systems

The danger in delivering new systems that access existing ones, is that organisations are perpetually building legacy systems.

The danger in delivering new systems that access existing ones, is that organisations are perpetually building legacy systems. Conventionally, they might access each of the parts of the legacy system directly, as illustrated in Figure 7. The problems with this are obvious and are further complicated when the integration code, such as protocol conversion and message formatting, etc., is embedded within the new application:

- Changes to, or replacement of, the legacy systems have a direct impact on the new application, causing double maintenance effort.
- Technical skills and domain knowledge relevant to the legacy systems are required to build the new system.
- No value is added to the legacy systems. Anyone wanting to build another new application has to repeat the integration exercise.

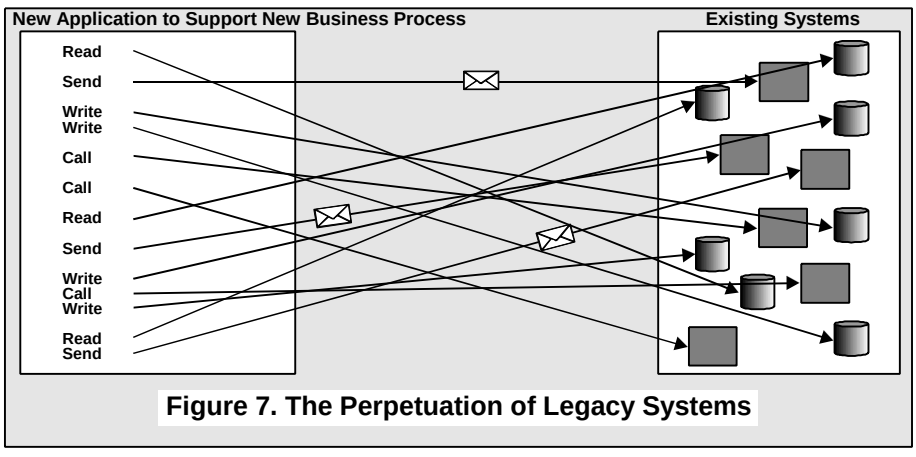

Figure 7. The Perpetuation of Legacy Systems

A more structured approach is illustrated in Figure 8. Whether the applications need to be integrated at the function, message, or data level is dependent on their current physical implementation. Wrapping can be used to turn all of these sources into components that offer a service through a common interface mechanism. Therefore, everything is logically integrated at the service level.

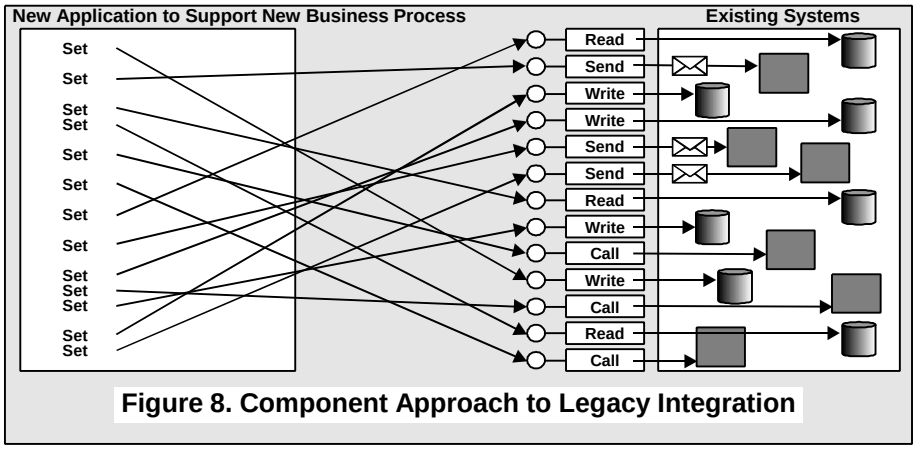

Figure 8. Component Approach to Legacy Integration

The benefits of this approach include:

- The new application is isolated, to a large extent, from changes to the legacy systems.
- The current physical implementation of the service is transparent to the consuming application.
- The skills to assemble the new application only have to be focused on the technology of the new application, not on the diversity of the legacy systems.
- The existing systems are now viewed as components and can, therefore, be assembled into a number of different applications without additional effort in terms of wrapping.

Adopting a more component approach like this avoids the new system effectively becoming part of existing monolithic applications.

Application Integration Architectures

It is our experience that integration can be perceived as being simpler than it really is. This is because many of the complexities of integration are not fully envisaged at the outset of a project. As suggested, Application Integration can be complex, with a danger that the resultant new applications consist of so much interwoven spaghetti code that they become very difficult to change in future. Butler Group believes that it is, therefore, important to take a more architected approach to Application Integration.

This section considers architectures that are appropriate to integration.

Butler Group's Integration Domains Model

Using layers is a common way of dealing with complexity and separating concerns into their proper domains. Our first model, illustrated in Figure 9, shows that integration needs to be considered at four key layers, and requires alignment, or some transformation mechanism, at each of the bottom three of this stack.

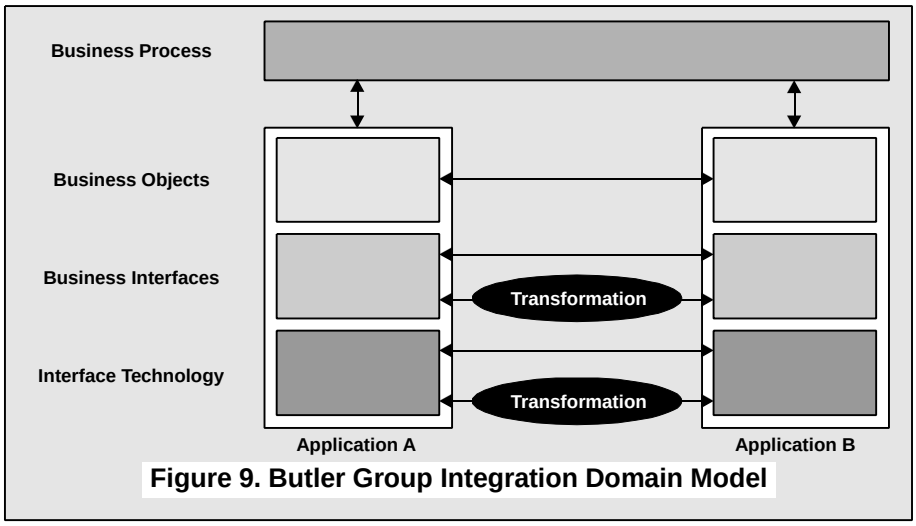

Figure 9. Butler Group Integration Domain Model

- **Business Process –** Integration requires an understanding of the sequence of events that triggers integration between components and applications, and the role each plays in the overall business process.
- **Business Objects –** Alignment of, or transformation between, the business objects contained in the components and applications. For example, exactly what is a customer, or a product, or an order?
- **Business Interfaces –** Alignment of, or transformation between, the interfaces and exchanges of information between the components and applications.
- **Interface Technology –** Alignment of, or transformation between, the technologies used to implement the business interfaces and achieve the integration.

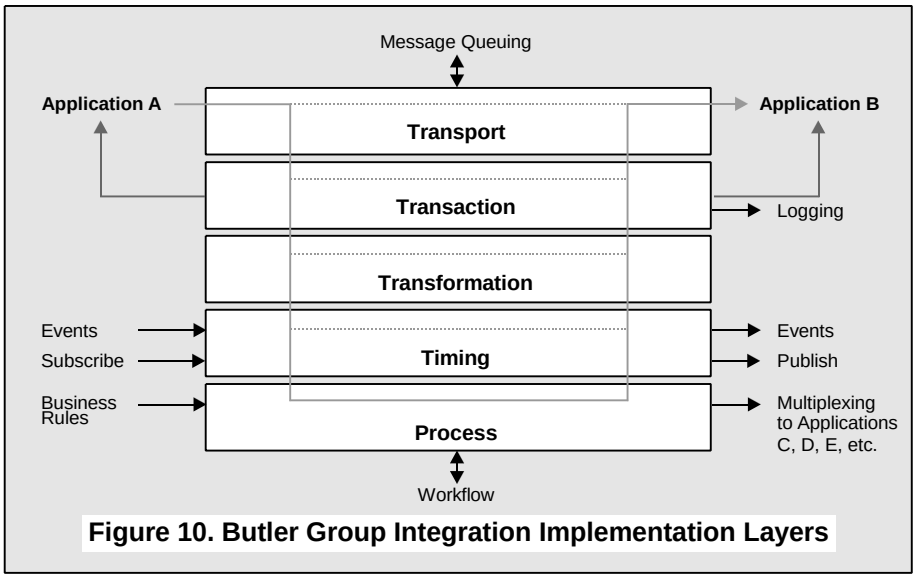

Figure 10. Butler Group Integration Implementation Layers

Butler Group's Integration Implementation Layers

Our second layered model of Application Integration (shown in Figure 10) is concerned with the detail of implementing a particular integration. This will help developers to fully assess the requirements for any particular integration project, by providing a check-list of functions that need to be considered in typical integration projects.

Transportation

The 'Transport' layer deals with the physical delivery of information.

The 'Transport' layer deals with the physical delivery of information, or connections between the sources to be integrated and the quality of delivery service:

- **Network Connectivity –** How and where is the connection established? How are differences in network protocols addressed?
- **Message Handling and Queuing –** Is the information message-based, and does it require the services of message queuing products? Is guaranteed delivery required?
- **Security and Encryption –** How are the different security requirements of each application or information source handled? Is a single log-on required? Should the messages be encrypted? Where does it take place?
- **Restart and Recovery –** What happens in the event of failure?
- **Routing –** Where does the message go? Are there rules governing the routing?

Transaction

The 'Transaction' layer deals with the integrity and management of transactions that involve integration between applications.

The 'Transaction' layer deals with the integrity and management of transactions that involve integration between applications. To ensure the real-time integrity of information, integration will also have to deal with the notion of distributed transactions and how, for example, two-phase commit, back-out and recovery approaches are handled. There are also issues of fault tolerance, for example, does information source 'A' stop recording new data if it cannot be sent to information source 'B' when the network is unavailable, or should off-line message queuing be used?

- **Transaction Management** – How are distributed transactions managed? How is transaction integrity ensured? How is state managed?
- **Fault Tolerance** – What mechanism is used to handle breaks in connectivity? How are faults traced?
- **Logging and Monitoring** – Are information exchanges to be logged? Is analysis of how integration is being executed required in terms of frequency and performance, etc.?

Transformation

The 'Transformation' layer deals with the conversion of information between the applications being integrated.

The 'Transformation' layer deals with the conversion of information between the applications being integrated. Although conversion of the data types and technical interfaces might be straightforward, handling differences in business objects and business interfaces between two applications is much more complex, and may even require information in further applications to be referenced in order to correctly format a message. Validation of the data might also take place in the transformation layer. Often, systems may store information in the form of coded values (for example, 'Customer Type' is stored as a single digit) requiring look-up tables to enable transformation:

- **Data and Protocol Conversion** – How are data types converted and protocol differences handled?
- **Semantic Conversion** – How are differences in business objects handled?
- **Message Formatting** – How is formatting and sequencing of data in the message handled? How are missing fields supplied?
- **Interface Conversion** – How are differences in business and technical interfaces resolved?
- **Data Validation** – What capabilities are there to validate data? Is just type checking or content validation required? Where does validation take place? Before the message is sent, during transformation, or on receipt?
- **Coding and Decoding Data** – How is data that is held in a coded form transformed?

Timing

The 'Time' layer deals with *when* information should be exchanged between applications.

The 'Time' layer deals with *when* information should be exchanged between applications. Events in one application typically trigger the need to send or request information from another. However, this could take place synchronously, perhaps requiring two-phase commit, or asynchronously via message queues. Events may have to be handled immediately for real-time action or queued for overnight replication. integrationware may monitor events in applications, or in the information that exchanges them, and fire off other dependent actions. There are various models of how the interaction between the applications should take place, for example:

- **Request and Reply** – 'A' requests information from 'B' and waits for reply.
- **Conversational (or Synchronous) –** Information is passed between two sources in a series of related exchanges. A reply must be received before processing continues.
- **Publish and Subscribe** – An information source publishes events that other anonymous information sources can subscribe to.
- **Asynchronous –** A reply to a message is not required before processing continues.

The time layer, therefore, provides services for:

- **Scheduling** – When should an information exchange take place?
- **Events** – What events should be monitored and which applications need to be informed?
- **Interaction Model** – Are synchronous and asynchronous activities supported? Are publish and subscribe techniques required?

Process

The 'Process' layer deals with the business rules that determine integration.

The 'Process' layer deals with the business rules that determine integration. More complex integration occurs when multiple applications need to be integrated, or when there are multiple steps in the integration process. Clearly, some integrationware architectures, such as 'Hub and Spoke', are better suited to these requirements than others.

Rules will also determine the integration. Rules will be applied in all the layers, for example, how data is transformed, or which application messages are sent to. However, there may be more complex business rules that need the content of information to be checked. For example, the type of customer may determine which customer information sources need to be updated.

Therefore, for more complex integration tasks, services are needed in the Process layer that address:

- **Multiplexing –** Are one-to-many and many-to-many interactions required? How are they managed?
- **Business Rules** – Can business rules that determine integration be expressed, and how? Where are they coded?
- **Workflow** – Is there a complete business process that must be executed to effect the integration? Is the integration a business process in its own right? Does the integration require multiple steps, with different sources integrated, to complete?

Some of these issues can be delegated to the operating infrastructure. Transport and transaction functions are commonly performed by the network, On-Line Transaction Processing (OLTP), Object Request Broker (ORB), or message queuing products that should be established in most organisations.

However, there will still be the requirement to integrate incompatible instances of these, so are an important part of the integration landscape. As such, it is common to see these services included within integrationware products, although frequently, this may reflect a 'bundling' of infrastructure services but this, at least, should ensure compatibility and that the total solution is in place.

Connection Architectures

Implementing this in an uncontrolled, ad hoc manner will lead to a complex Web of connections that is difficult to unravel.

In any organisation, there are likely to be a large number of connections to be made between different applications needing to be integrated. Common business infrastructure applications will often be integrated into many new applications. Implementing this in an uncontrolled, ad hoc manner will lead to a complex Web of connections that is difficult to unravel when change next occurs. Common connection architectures are:

- **Point-to-Point** – Applications are simply connected directly to one another. The routing, etc. is fixed between the two applications.
- **Hub and Spoke** – Applications are connected to a hub. The hub contains the rules for connecting applications together and marshalling messages between them.
- **Publish and Subscribe** – Applications that make information available, have no idea what other components use this information. Suitable for broadcasting information where no coupling, or even a response, is anticipated. A variation of this is a 'Bus' architecture, where messages are put on a logical bus and are not required to pass through a central hub.

Reducing the number of connections will clearly reduce the complexity. The 'Hub and Spoke' approach to reducing the number of connections between points is well understood, and popularised by the airlines. The Hub and Spoke approach is clearly an elegant architecture that both manages complexity, and makes future requirements easier to satisfy. As each new application is introduced to the hub, all others can make connections with it straight away. Integration between applications, that are already connected to the hub, can be quickly defined in the hub.

Organisations need to balance the needs of integration when selecting a connection architecture.

However, organisations need to balance the needs of integration when selecting a connection architecture. Whilst 'Hub and Spoke' might be the most flexible, it can introduce unnecessary performance overheads, as each message is interrogated and 4GL rules interpreted only to send it to the same place the previous 10 million were also sent that day. 'Publish and Subscribe' provides another form of flexibility, in that it requires no integration effort on the part of the subscribing application, but it might not handle the complexity of rules or a diversity of different technologies.

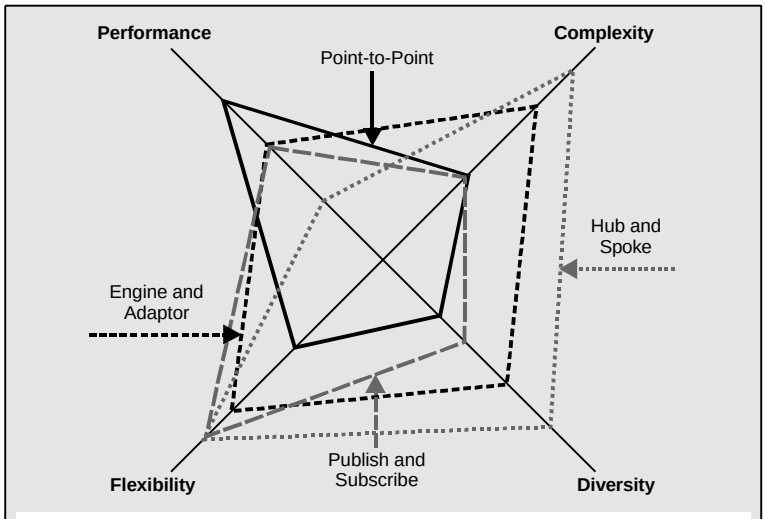

Figure 11. Balancing the Needs of Connection Architectures

Given the diversity of requirements and applications, there is no connection architecture, and most organisations will require a mixture. As illustrated in Figure 11, the appropriate connection architecture will be determined by a number of factors, including:

- **Performance** – Is a high-performance link required?
- **Diversity** – How many different types of information sources need to be integrated?
- **Flexibility** – Will integration sources and rules need to be changed frequently?
- **Complexity** – Is the integration simple or complex?

Engines and Adaptors

A further architectural refinement is to separate out the basic engine of integration, from the specifics of the particular applications in each instance of integration. This approach of engine and adaptor is common of integrationware, enabling them to integrate multiple alternate message formats, protocols, interface technologies or packaged applications with a single product.

Adaptors are commonly available for popular packaged applications, databases, platform technologies, such as ORB or OLTP, message queuing and popular message formats. An increasing number of products are also providing a Software Development Kit (SDK) to enable adaptors to be built in-house for legacy applications, or by smaller Independent Software Vendors (ISV) for less common packages. Hub and Spoke integrationware typically involves an engine and adaptor approach.

Engine and adaptors help isolate against change, as only a new adaptor is required when a component in the integration is swapped for another.

Engine and adaptors help isolate against change, as only a new adaptor is required when a component in the integration is swapped for another (as illustrated in Figure 12). On the downside, they can introduce a performance overhead, though Butler Group expects more products to also support a compiled approach to minimise this (at the expense of more dynamic flexibility).

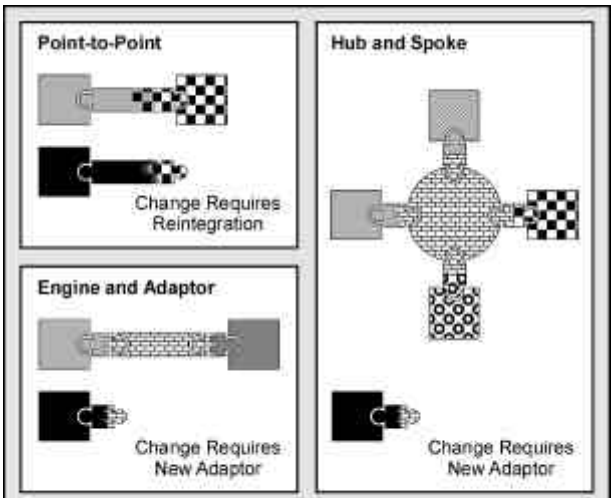

Figure 12. Engines and Adaptors

Layered Application Architecture

Integration Standards

In the long term, efforts to establish common architectures on an industry basis promise to greatly reduce the integration challenge by providing industry standard business and technical architectures.

Clearly, many of the challenges of integration, particularly the needs for transformation, are reduced when applications share compatible business and technical architecture. In the long term, efforts to establish common architectures on an industry basis promise to greatly reduce the integration challenge by providing industry standard business and technical architectures. However, it is clear, at least with our current horizon, that this will not solve the problem entirely. Firstly, there are likely to be multiple competing standards within the same industry, so unless the entire industry truly accepts a standard, issues of dealing with trading partners, or mergers and acquisitions will work against any enforcement of a standard within a single organisation. Secondly, the huge installed base of legacy applications and packages will not conform to such standards, and would require significant effort to upgrade or replace them.

If organisations cannot standardise the architectures for the implementation of the applications, one compensation might be to establish standards for how they should be integrated. Butler Group observes a surge of interest in standards that enable integration. Virtual supply chains and e-business are also driving integration standards, in recognition that, by their very nature, they cross many organisations, and cannot dictate the implementation of applications or components that provide the services required by the business process. The following initiatives, though not always setting true *de jure* standards, recognise that agreement is required, at least on how data is exchanged or what role each application should play in a wider business process, and are particularly relevant to enabling integration at a level above the pure technology layer.

However, each of these standards, typically, only addresses part of our integration layers, and sometimes overlap. So, for example, whilst eXtensible Markup Language (XML) might solve many of the transformation issues, it does not address the other integration layers. Additionally, they still require the existing base of applications be upgraded, in order to make connections to them using these approaches. Also, despite recognition of the need to implement new solutions for e-business, the ubiquity of the Internet and its reach into every level of organisation, there is considerable investment by larger organisations in existing standards such as Electronic Data Interchange (EDI).

Despite concerns about the contradiction of multiple standards, Butler Group believes compliance with these standards will be useful, even where there is no obvious requirement to interface to external organisations today. Remember the quotes from Tom Peters, just because a service is provided in-house today, does not mean it will be tomorrow. Additionally, you will be increasingly likely to find packaged applications and components, plus ready built integrationware adaptors, that already comply with these standards and enable conversion of your applications to them.

The following represents some of the standards and initiatives for integration that should be considered by organisations:

- **OAGIS** – The Open Applications Group was formed some years ago by several major package vendors who recognised the need to provide interoperability between their packages, and has seen a doubling of its membership last year, reflecting the increasing attention on integration. Its Open Applications Group Integration Specification (OAGIS) provides standard message formats in the form of Business Object Documents (BOD) that reflect common business exchanges.

- **IMWA** – The International MiddleWare Association (formerly MOMA), which is working to enable interoperability between messaging products.
- **XML** – eXtensible Markup Language. XML removes the need for data transformation. XML, in particular, shows every indication of universal adoption. Several of these integration initiatives include XML, for example OAG, Microsoft's BizTalk, and RossettaNet, who are attempting to set establishing business semantics and process standards on top of XML technology.
- **UCC** – The Uniform Code Council has been long established in defining product codes and barcode standards to enable electronic communications through the supply chain. It is now also focused on defining the relative business processes.
- **EDI EDIFACT, X12** – Electronic Data Interchange. EDI is one of the most mature and established integration standards, with EDIFACT and X12 widely adopted for exchange of data between organisations. EDI defines several hundred standard maps of data that reflect common business exchanges.
- **Industry Standards** – Several vertical industries have established their own standards for data exchange. For example, HL7 in Healthcare, S.W.I.F.T. for financial transactions, or RossettaNet for IT.
- **RossettaNet –** RossettaNet is a good example of the sort of initiative that is likely to start in many industries to solve e-business integration problems. RossattaNet is focused on the IT product supply chain, and was recently established by a who's who of IT companies, together with end-user organisations, in their supply chain (such as Fed-Ex and UPS) to define business processes, business objects and interfaces to enable e-business integration.
- **Microsoft BizTalk** – Microsoft recently announced BizTalk, an e-commerce framework based on XML, and vertical industry standards to enable organisations to integrate applications and conduct business over the Internet with their trading partners and customers. It defines business schemas for corporate purchasing, product catalogues, offers, promotional campaigns and other business processes.

At a technical level, Butler Group expects a wider adoption of interface mechanisms provided by technologies such as COM, CORBA, and Enterprise JavaBeans, plus rapid adoption of XML, to simplify integration, though there is likely to be equal adoption of them all. As they can deal with multiple programming languages, plus provide wrappers on to legacy applications and packages, they can at least reduce the number of variations that developers face when faced with integrating applications with diverse and proprietary interfaces. The broad adoption of Internet technologies is also likely to simplify integration at the technical level.

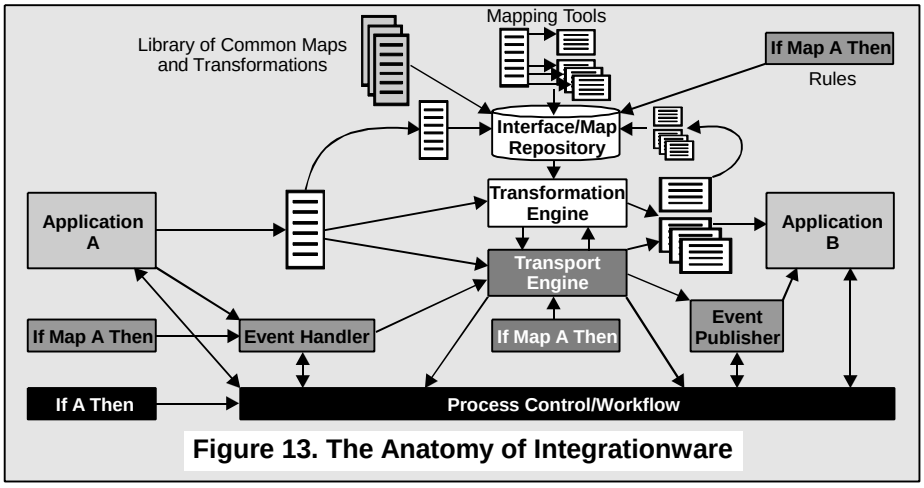

Figure 13. The Anatomy of Integrationware

Integrationware

Anatomy of Integrationware

The functionality of integrationware is illustrated in Figure 13. However, not all products will have all this capability and it is more common that a selection of integrationware tools are needed. integrationware consists of the following main elements:

- **Transport Engine** – The mechanism required to transport messages from one application to another. This may not be required in call mechanisms as the connection is established by the network, but is common in messaging connections. It is frequently provided by the operating infrastructure, but is frequently considered part of Application Integration as this is its common, but important function. Could be provided simply by the network, or by message queuing products, message brokers or object request brokers, or combination of them.
- **Transformation Engine** – A key role of integrationware is the transformation of information between applications. Modern transformation engines consist themselves of several elements.

  Mapping Tools – The means to express, often visually, the mapping of data between one format and another.

  Repository – A library where maps and interfaces are stored, enabling reuse.

  Library of Common Maps and Interfaces – The interfaces to common packages or technologies, together with maps of frequent integration between them, are often supplied with integrationware.

  Engine – Performs the actual transformation, taking the maps as input. It might do this dynamically, or via some mechanism to compile the maps, improving performance.
- **Event Handling –** As well as 'Publish and Subscribe' mechanisms, integrationware products might also offer the ability to monitor events and trigger integration flows in response. Events might be time-based, the arrival of a message, or a change to a data record (perhaps using triggers). Sometimes referred to as watchers, or listeners, these can be useful to enable integration from self-contained monolithic applications that were not originally designed to notify the outside world of their activities. An additional complexity is when multiple events must be monitored.
- **Process Manager** – A mechanism to define the sequencing of tasks, or workflow of the integration, often using a visual modelling approach. In recognition that integration is often a multi-step process, some integrationware also include ready-built business processes, often delivered in a workflow environment. These include pre-defined plug points, into which existing applications and packages are connected at appropriate points in the process. This is sometimes referred to as 'Processware'.
- **Rules** – One of the complexities of integrationware is that each of the above elements can be rule driven. Content-based rules can define the routing of messages in the transport engine, or the transformation of data. The event handling is likely to contain rules, and the business process most certainly will. Development and maintenance problems will arise when these rules are dispersed across several integrationware products and coded in different languages.

As discussed earlier, Application Integration products are frequently delivered as a core engine, and a set of pre-built adaptors that contain common transformations and connections. Also, the development environment is often separated from the run-time integration engine. However, an increasing need to implement and change integration in a real-time basis, means that this is not necessarily always desirable. Rather, a combination is required with, perhaps, some compiled adaptors built off-line to maximise performance, whilst others being interpreted at run-time.

When to Use Integrationware

The degree of complexity involved in Application Integration will vary enormously. Butler Group characterises four levels of integration complexity:

- **Plug-and-Play –** Integration between applications that share common business objects, business interfaces and technology.
- **Well-Defined** – Integration between applications with well-defined, though incompatible objects and interfaces, for example, Package-to-Package.
- **Undefined** – Integration of in-house legacy, and new in-house build with each other or with a packaged application.
- **Complex** – Multiplexed integration of multiple sources with complex rule-driven integration.

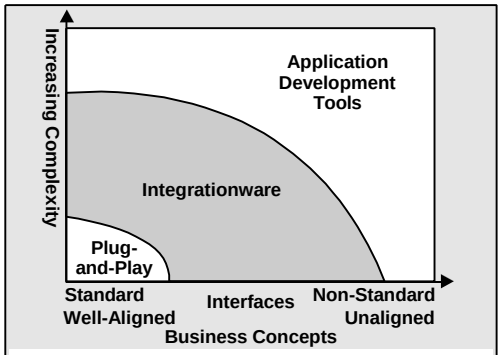

Figure 14. When to Use Integrationware

At the plug-and-play level, the role of integrationware is diminished. When applications to be integrated share a common set of business objects, interfaces and technologies, then there will be no requirement to use integrationware products for transformation, although a transport engine may still be required, for example, for message queuing, though this might be provided as part of the operating infrastructure. Application Integration is still required in the form of assembling application components into a new business process, and integrationware in the form of business processware will still be useful.

integrationware is most applicable, and more likely available, for combinations of the disparate applications that have well defined, but incompatible business objects and interfaces, and are also the most commonly implemented across organisations. The behaviour of packaged applications is clearly the same across organisations providing a ready well-defined market. integrationware can enable Application Integration at almost plug-and-play levels, when connecting application packages by automating the connections and transformations.

However, in a large number of situations, there will be no well-defined behaviour or interfaces available, at least at one end of the connection. A significant cost of package implementation is often legacy integration. The best that integrationware vendors can do, is to provide a SDK to build adaptors for your legacy applications. Even though integrationware is offered for common platform technologies that address the plumbing, or wiring level of integration (for example, adaptors to integrate CICS, CORBA or COM), differences in business semantics between the package and legacy will remain.

Enterprise Application Integration (EAI) vendors guessed that organisations who have bought packages, would probably also be disposed towards implementing Application Integration off-the-shelf too.

When integration requirements are not readily satisfied by off-the-shelf integrationware, and the integration rules are complex, the task is effectively application development. Even so, some of the more sophisticated integrationware products are able to support this, although Butler Group suspects that they may, themselves, have been called 'Application Development' tools had not 'Application Integration' become a more fashionable term.

Why Use Integrationware?

Butler Group observes the following key reasons to implement integrationware:

- **Providing flexibility and Adaptability to Future Requirements** – Use of integrationware and appropriate architectures can prepare applications for unforeseen integration needs, and enable them to be re-integrated with a minimum of effort.
- **Reduction of Development and Maintenance Effort –** Using commercial integrationware, rather than in-house development, can significantly reduce developer effort and avoid constant "re-invention of the wheel" in each integration project.
- **Resolution of Skills Issues** – The diversity of applications, both technically and from a business semantics view, means there is bound to be a shortage of developer skills who can deal with all of the differences. integrationware can reduce the number of alternatives that need to be dealt with by hiding the applications implementation behind standard interfaces.

The Integrationware Market

As with every craze, a huge number of software vendors are suddenly climbing aboard the Application Integration bandwagon.

As with every craze, a huge number of software vendors are suddenly climbing aboard the Application Integration bandwagon, relabelling their products and claiming to be a leader. As a consequence, there appears to be a bewildering diversity of products offered as the solution to integration challenges.

In reality, each of them has a role to play in Application Integration. Some of them are, in effect, part of the operational infrastructure and technology necessary to both implement and integrate applications such as middleware and messaging products. Many of them are dependent upon each other to enable integration and, as such, are frequently being bundled together by the vendors to provide an integration solution. As a result of this organisations need to be cautious about vendors claims of product 'X' being better than product 'Y', when, in fact, both are revealed to contain the same third-party elements. Nevertheless, knowing that the vendor is likely to have ensured that the parts of the bundle work together, is much easier than trying to integrate the integrationware yourself.

Butler Group observes that there will be a rapid consolidation of integrationware from the following directions:

Enterprise Application Integration (EAI)

The first wave of integrationware vendors took a more 'solution-oriented' approach. So called EAI vendors guessed that organisations who have bought packages, would probably also be disposed towards implementing Application Integration off-the-shelf too. Their engines are typically rule-based, enabling complex decisions to drive the integration and transformation, not just simple 'A' to 'B' connections. Some recognise that integration takes place within the context of a business process, and include ready-modelled solutions to drive integration.

The issue is one of coverage. Most EAI vendors provide adaptors for leading ERP packages, including SAP, Oracle and PeopleSoft, and customer interaction packages, including Clarify, Vantive and Siebel. Significant players, such as JD Edwards and Baan, are clearly next on many vendors lists. Platform integration is, most commonly, supporting MQSeries, CICS, CORBA, COM and the leading databases. Beyond these mass market requirements you have limited choice or are on your own, though nearly all EAI vendors supply adaptor 'development kits', which is still potentially preferable to building your own point-to-point solution, especially if you are going to be using a particular integrationware environment for some fully supported environments.

EAI approaches can be considered when:

- The prime need is for application package integration, especially Package-to-Package.
- It, perhaps, requires minimal support of central IS staff.
- Information sources are already well structured and have clear interfaces.
- The development of new functionality is already well supported by other tools.

Application Servers

The last year has seen an explosion in the announcements of application servers. These primarily provide the infrastructure support for executing distributed applications together with technology integration capabilities. On their own, they provide little in the way of adaptors beyond the ability to achieve integration at the technology layer by including access to common technologies, such as COM, CORBA, JavaBeans, C++, databases, ActiveX components and some legacy environments. Although third-parties are, or will undoubtedly be, supplying additional adaptors for the most popular servers. Butler Group also expects considerable partnering and bundling of integrationware with application servers. An application server can be considered when:

- The prime need is to deliver Web-based business processes, with reasonable development of new functionality.
- The focus is technology integration and legacy systems.
- Sources of integration are already well structured and have clear interfaces.
- When developing new components that run on the Web server.
- Providing a Hub and Spoke mechanism.
- Providing run-time management of applications.

Application Development Tools

Many developers will ignore commercial integrationware, and simply write code.

Many developers will ignore commercial integrationware, and simply write code. 'Class' libraries and SDKs that enable technology and application package integration to be available as add-ons for many of the Object-Oriented (OO) 3GL programming tools. Several development tools, notably those with a 4GL background, also include ORB technology that can provide interfaces to many alternative interface technologies, and provide wrapping to transform them. Most of the 4GL/Code generator development tools are now providing wrappers so that their users can unlock the services within their otherwise proprietary applications, and make them more easily accessed via component technologies, such as COM. This can take the form of a client/server application, where the client has no User Interface (UI) but exposes component interfaces to enable integration.

Although they are not usually promoted as integrationware by the vendors, the multi-platform capabilities of 4GL/Code generators, combined with these, component wrappers can provide a useful mechanism for building wrappers, particularly when those wrappers need to include functions beyond protocol conversion and message formatting, for example, merging and replication, or extension and exclusion functions. Most of them have the ability to call external sub-routines that may allow non-invasive wrapping. 'Application Development' tools can be considered when:

- The prime focus is on building the new applications that require integration of existing applications.
- There is requirement for integration of different technologies.
- There is complex rule-based integration.
- The Applications to be integrated were developed in same tool.
- The existing applications must be re-coded to enable integration.
- Developing new components for a subsequent integration.

Screen Wrapping tools

Screen wrapping (or scraping) tools have evolved from simply enabling a Graphical User Interface (GUI) veneer for client/server applications, to now also providing component/object interfaces to encapsulate host/terminal applications, and enable their integration into new application workflows. Wrapping can be considered for:

- Host/terminal applications that have no other interfaces beyond screens.
- Existing applications that cannot be, or it is undesirable to, re-engineer to enable better integration.
- Providing object interfaces, for example, COM, CORBA and JavaBeans, onto existing host applications.

Workflow

We expect Application Integration to increase attention on workflow. Workflow engines are particularly adept at defining more dynamic processes that require change in real time, perhaps by users themselves. Workflow can be considered:

- When developing new business processes.
- When developing dynamic, rule-based integration.

Middleware

Application Integration has always been a strong theme for middleware vendors, though, in the past, the focus was perhaps more upon integration within an application, rather than between them, with some form of middleware layer always used when building distributed applications, and assembling a collection of objects or components into an application. Messaging and object request brokers are equally applicable to Application Integration, as they provide complimentary functions and both play a strong role in the technology layer of integration.

Messaging Middleware

Messaging can be considered when:

- Implementing loosely coupled applications.
- There is the need to integrate applications between organisations where the availability of applications cannot be fully guaranteed.
- You require guaranteed delivery of messages between applications.

Object Request Brokers

ORB products can be considered when:

- You need tightly coupled applications.
- The sources of integration are already well structured and have clear interfaces.
- Implementing architectural standards.
- Developing new components for subsequent integration.
- Wrapping existing application interfaces into a single technology.
- Integrating in-house applications.

Database Gateways

Though we have stressed the increasing importance of encapsulation of applications being integrated, there are occasions when direct access to the databases is the only option. They are, of course, useful to build the wrappers to encapsulate existing data sources, making them easier to integrate. There are multitudes of database gateway products that can simplify access to diverse data sources. Database gateways can be considered when:

- The interface offered by a database management system is the only practical option.
- Building wrappers around existing data sources, to enable integration via encapsulated interfaces.
- Integration is not real-time (for example, replication and batch loads).

Package Vendors

Package Vendors are not normally vendors of integrationware per se, though they may often bundle it with their applications as a means to simplify integration. Package Vendors are the sources of business objects and business interfaces that require integration. The pressure is now on Package Vendors to provide the published, stable interface mechanisms to better enable integration in the first place. However, the need for organisations to upgrade to the latest versions of packages that support these interfaces will slow the uptake, as the business case to upgrade, just because of the interfaces compared to the significant effort to do so is not clear, even though the desire for integration is, itself, the justification.

Selecting Integrationware

Organisations should balance immediate requirements with long-term integration needs.

Selecting integrationware is the usual following combination of features and vendor viability. Organisations should balance immediate requirements with long-term integration needs, and look at what additional packages and technologies are supported by particular integrationware products beyond their current requirements. Vendors with a broad support are now likely to continue that trend.

- **Fit for Purpose** – What Application Packages, interfaces and message formats, and technologies are supported? Which integration layers are supported? *[See Table 2 Typical integrationware Functions for an indication of the typical functionality coverage of different integrationware products.]*
- **Adaptability** – How easy is it to adapt integration to changes in business or technology? Are you trying to put a long-term architecture in place, or just trying to solve a point-to-point problem?
- **Ease of Use** – What developer skills and resources are required? How do the tools function? Is visualisation and diagramming used, or is it purely text/code driven?
- **Vendor Profile** – What is the financial viability of the vendor? What partnering and distribution channels are in place, and what support and services are offered?

Integrationware	Business Process	Business Objects	Business Interfaces	Technical Interfaces	Transport	Transaction	Transformation	Timing	Integration Rules
EAI	Y	T	T	N	N	N	Y	Y	Y
Application Servers	N	N	T	Y	Y	Y	N	Y	D
Messaging	N	N	T	Y	Y	Y	3rd	L	D
ORB	N	N	N	Y	Y	Y	N	Y	D
Workflow	D	N	N	Y	Y	N	N	Y	D
Screen Wrapping	N	N	T	Y	N	N	Y	Y	D
Database Gateways	N	N	N	Y	Y	N	N	N	D
Development Tools	D	D	T	Y	Y	Y	D	Y	D
Packages	Y	Y	Y	N	N	Y	Y	Y	Y

Key: Y-Usually provided, N-Not usually supported, D-Enables development of, T-Transformation of, 3rd-Include third-party.

Table 2. Typical Integrationware Functions

Market Assessment

Given the size of the problem, and the inevitable demand for rapid response to application backlogs post Year 2000 (Y2K), there is clearly a huge interest in this area.

Given the size of the problem, and the inevitable demand for rapid response to application backlogs post Year 2000 (Y2K), there is clearly a huge interest in this area. Butler Group expects to see a dramatic increase in integrationware during the period 1999 to 2002, as a practical and timely answer to business problems which will not wait for industry standard interfaces to simplify the problem. However, whilst package vendors may have an imperative to provide more standard interfaces to encourage adoption of their products, integrating in-house legacy systems will remain a major challenge for IS organisations. Additionally, there will continue to be multiple 'standards' in each area and requirements to bridge them.

The relevance of each of the different types of integrationware is illustrated in Figure 15. integrationware at the bottom of the stack is effectively operation infrastructure, whilst those in the middle are relevant to the development to applications through Integration, There will be increasing commoditisation towards the bottom of the stack, as their integration functions become undifferentiated, although their broad applicability will provide a viable market for many vendors. However, those at the top are of higher value to the business, and will continue to be differentiated as new business processes are identified and implemented in integrationware.

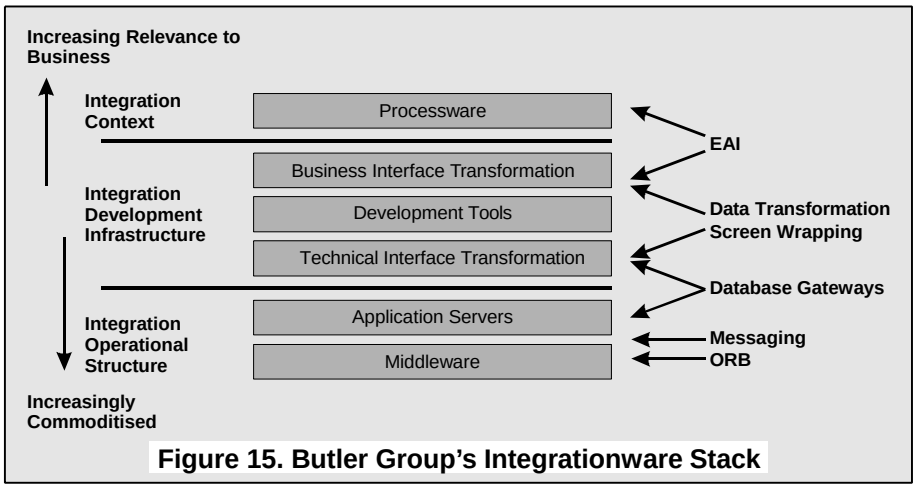

Figure 15. Butler Group's Integrationware Stack

Market Directions – The Future For Application Integration

Market Consolidation

Butler Group would, therefore, expect relatively rapid consolidation to occur, particularly as some of the bigger software companies decide to muscle in on this lucrative area.

Many integrationware companies are small, and the investment that is required of them to support the diversity of requirements is significant. Butler Group would, therefore, expect relatively rapid consolidation to occur, particularly as some of the bigger software companies decide to muscle in on this lucrative area. Some of the smaller integrationware vendors will be acquired by the package and infrastructure vendors, others by the large software conglomerates, whilst many of the remainder will merge to provide a broader integration capability. This is also giving rise currently to a considerable number of partnering and reselling initiatives between the vendors, as they each try to increase the coverage of their offerings. Larger vendors, in particular, will move to make sure they have the whole of our integrationware stack covered and, hence, offer a one-stop shop for their customers.

Also, as part of the natural evolution, we would expect to see some sector specialisation, with integrationware vendors gaining prominence in particular areas by addressing specific needs. There is clear opportunity for specialisation in vertical industries, for example, finance, telecoms, insurance and retail, etc., by supporting business processes, standards and packaged applications that are industry specific. The opportunity for specialisation decreases for packaged applications, and even less for technologies, as most vendors will be expected to support a wide variety.

Butler Group also expects leading development tools to include more Application Integration capability, particularly to address common package integration requirements, business and technical interface transformations, and dynamic workflow/process capabilities.

There is also an opportunity for modelling tool vendors to more explicitly support Application Integration. Most modelling tools are currently biased too much towards green field new build. Application Integration can benefit greatly from visualisation, and integrationware tools often feature visual transformation mapping and workflow diagrams. Modelling tools that can reverse-engineer the interfaces, data structures and business objects of existing applications, plus model new workflows and components, and document the integration requirements, connection architectures, and perhaps generate the transformation required to bring them all together, would be a useful addition.

Increasing Integration Market

Given there are, for example, several thousand SAP implementation projects alone underway at the moment, and the EAI vendors measure their customer-base in the hundreds, it is not hard to see the opportunity for growth.

Butler Group believes the market for integrationware is set to significantly increase. Given there are, for example, several thousand SAP implementation projects alone underway at the moment, and the EAI vendors measure their customer-base in the hundreds, it is not hard to see the opportunity for growth. In addition, as many of the legacy systems will prove too difficult a problem for integrationware products to integrate through automation alone, there is also a significant opportunity for services. This last point is not lost on the Systems Integrators, who are building integration support as quickly as they did ERP practices. Given it is widely accepted that integration costs significantly more than the package, choosing Systems Integrators on their ability to provide integration services is clearly just as important as their domain knowledge in the package being implemented.

Whilst, overall, the demand for integration will rise in response to the need for rapid delivery of new applications, wider adoption of standards will ultimately reduce the need for transformation integrationware. Commoditisation is also likely to occur in the transformation and technology layers as differentiation, or the requirement erodes.

However, the constant introduction of new business initiatives will see a steady growth in integrationware that provides ready-built business processes. Also, a backlash against large monolithic packages that prove inflexible to the next round of changes post Y2K, will see greater acquisition of more fine-grained components and frameworks, which are becoming available at the same time. There will also be a separation of the acquisition of the business process, which defines the application, from the business objects or components that provide the services required by the application, again increasing the need for integration.

The Impact of Standards

The need for integrationware vendors to supply a wide variety of adaptors, is a reflection on the proprietary access mechanisms of the packages and legacy applications they must integrate.

Whilst Butler Group expects the wide adoption of standards by application package vendors, to enable interoperability at the technology level, more challenging for small-to-medium package vendors will be the issue of standards for business objects and interfaces. These are more likely to be set in *de facto* fashion for each vertical industry by the relative dominant package vendors, such as SAP, or Microsoft's vertical business initiatives. Adoption of standards at all levels by the package vendors will be less rapid than hoped, due to the significant effort needed to re-engineer applications to support them, and the probable requirement for many package vendors to support several competing alternatives.

The apparent contradiction of many standards will still be less of an issue for end-user organisations than the profusion of proprietary interfaces are today, and integration between applications that support common standards will be largely transparent thanks to integrationware.

Ultimately, due to the adoption of standards, the current lack of available adaptors from a particular vendor will not be such a constraint in the long-term as it might appear. Even so, whilst some of the transport and transformation issues might disappear, the automation of the other aspects of integration and the provision of business processes that these products supply, will still be welcome. And, of course, the enormous requirement to integrate legacy applications will not be addressed by these standards, as long as their proprietary interfaces remain.

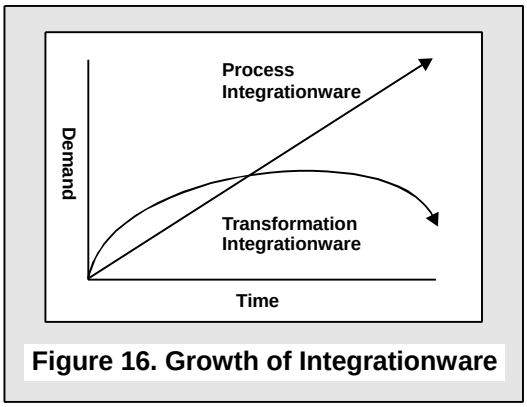

Figure 16. Growth of Integrationware

Butler Group, Europa House, 184 Ferensway, Hull, HU1 3UT, United Kingdom
Tel: +44 (0)1482 586149 Fax: +44 (0)1482 323577

WWW.BUTLERGROUP.COM