Why worry about integration?

Richard Lancaster
LANSA Product Center

Integration is a complex task spanning networks, hardware platforms, operating systems, programming languages, databases, applications and business entities. Integration is not a plug-and-play device. It involves multiple business processes and integration points, collaboration across functional boundaries and numerous technologies.

Integration is a necessary component for managing information when you operate more than one application. However, integration will become just another large and complex application, unless implemented as a component in an overall information management solution.

The essential ingredients of efficient and effective integration are:

- **Flexibility** – loose coupling for the components to facilitate change and maintenance.
- **Simplicity** – an integration bus to reduce the number of point-to-point integration occasions.
- **Appropriate fit** – the right integration technology and tools for the job.
- **Part of the family** – integration solutions do not operate alone; they are part of the solution.

LANSA products play a role in integration projects by managing the integration, or by collaborating with specialist integration tools such as an Enterprise Service Bus (ESB).

What Type of Integration do I need?

The characteristics of an integration solution depend on the nature of the business requirements. What are we integrating – information and/or processes? Why do we need the integration? Answering these questions requires analysis, evaluation and decisions. Below are some issues and questions to consider when faced with an integration challenge.

Business drivers – Such as company initiatives, regulatory responsibilities and compliance obligations.

Requirements – Will highlight the project scope and the likely effort to build and implement. Consider the requirements from the perspective of a company wide integration strategy.

Scope of the integration – Is the integration part of a corporate strategic initiative, or is it tactical? This decision may influence the nature of the solution and its budget.

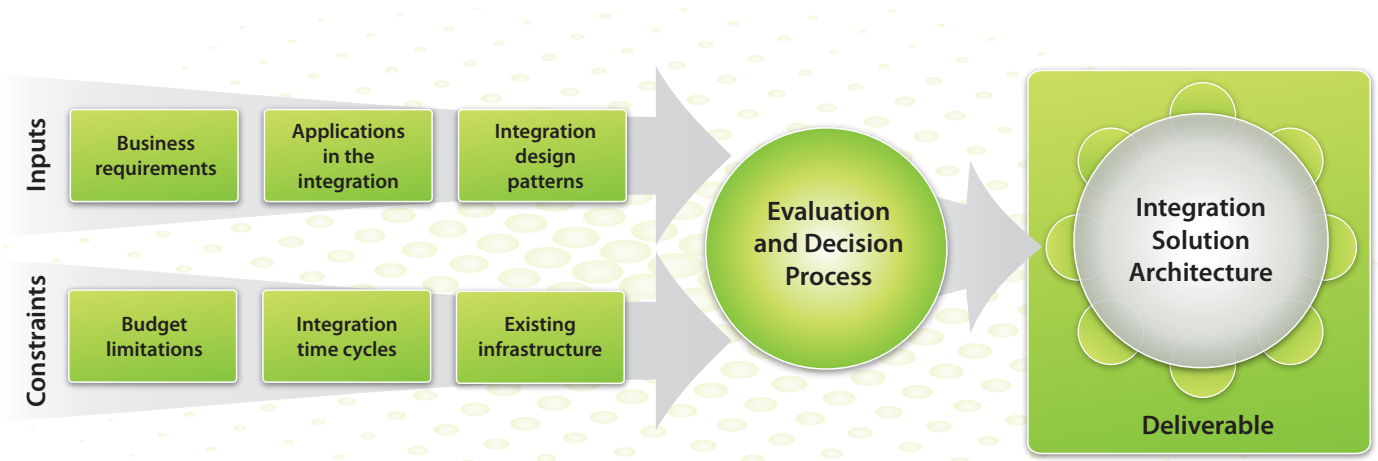
Cost benefit and return on investment – When calculating the costs and benefits for an integration project ask these questions: What are the measurements for determining success? How to collect the measurement data? How to determine the success or otherwise for the project?

Automated integration – Can you describe the business rules to the integration tools? Is it practical to delegate all decision making to automated processes? If the process requires any human intervention, a fully automated integration is impractical. Analyze the integration process and determine the parts suitable for automation.

Semi-automated integration – Semi-automated integration solutions have some parts of the integration solution managed manually and other parts automated. This type of solution is usually a compromise constrained by cost or the need for human intervention in the process. If it is possible to eliminate the human intervention, determine the implications of a fully automated solution.

Integration techniques – Identify the integration techniques needed for the solution – messaging, data mapping, data transformation, business process management, Web services and portals.

What applications are involved – The nature of the applications will influence the architecture for the integration solution and influence the level of automation. ➔



What Factors Influence the Design?

Having examined the factors that influence the integration project, the next step is to design the integration solution. Choosing the appropriate solution is the objective. Building a loosely coupled, message-based integration solution might represent best practice for a flexible, extensible and scalable integration solution, but it will be overkill when a file transfer is all that is needed.

The following list presents topics to consider when designing integration solutions.

Application coupling – Minimise dependencies between applications to allow each application to evolve without constraint. Loosely coupled integration solutions will be more flexible, but may require replacing applications that exhibit a high dependency on each other.

Data formats and transformations – Applications use data in different formats. Determine the rules for reformatting and transforming data between applications.

Design integration solutions for business processes – At the technical level, integration deals with interfaces between applications and databases. However, the reason for integration is to streamline business processes. The focus should be on the business outcomes. This will help you to see the wider perspective and not just an interface between two applications.

Discourage hardcoded integration – Hard-coded integration will prove less flexible and cost more to maintain than integration built with integration tools (for example an ESB).

Intrusiveness – Minimise the changes to individual applications and the code needed to manage the interfaces. This may involve compromising the optimum integration methodology. Above all other factors, you need the flexibility to replace an application in an integration solution with a minimum of effort and disruption. Changes to application code reduce flexibility and increase the cost of building and maintaining the integration solution.

Plan for incremental deployment – Large deployments exhibit greater potential for something to go wrong.

Reliability – Monitoring and notification are key ingredients in a reliable integration solution – knowing what is going on will help avoid problems and speed up their resolution. Reliability in this context refers to the whole integration solution and assumes reliability in the underlying infrastructure (networks, hardware and operating system software).

Share data, share functions or share both – Data sharing is appropriate when you need to be certain of the data integrity. With one source for the data, you know how and where data is maintained. Data sharing may be the only viable option when an application cannot share its functions. Function sharing enables reuse for code. A working function with a history of correct results requires less maintenance than writing code from scratch.

Synchronous or asynchronous – For synchronous calls, the calling application will wait and do no work until the called application responds. Synchronous calls will experience longer response times when distance or network reliability extends the loop delay. Asynchronous calls are more efficient but these solutions are more complex to design, build, debug; and managing exceptions can be problematic. Use asynchronous calls when the integration will tolerate latency.

Timeliness – Timeliness means determining the appropriate timing cycle for transmission. Information integration may tolerate latency but synchronous function integration must occur in real-time. Timeliness for one application may be inconvenient for another application. This becomes even worse in the case of an integration solution built on a publish-subscribe model. For example, one application may publish data to several applications, all of which apply different rules for the timeliness of the data.

Build the integration solution as a set of services – This will provide granularity for the scope of an integration function and promote reuse. Discrete services are easier to replace and maintain.

Selecting integration tools – Buying integration tools will produce a faster implementation than doing it all from scratch. However, developers may need specialised training and you may suffer vendor lock-in. Building the integration from scratch will involve a longer leadtime and, unless you get the architecture right, may require more maintenance.

How Does LANSA Help Me with Integration?

Most LANSA products will help integration projects, by themselves or in collaboration with specialist integration software. The following examples suggest the LANSA products to use for different types of integration projects. ■

LANSA Products	Integration using Shared Databases
LANSA Integrator	RPG/COBOL and C#.NET programs updating a shared database.
LANSA Open for .NET	RPG/COBOL and C#.NET programs updating a shared database residing on an IBM i server.
Visual LANSA Framework	Visual LANSA and C#.NET programs updating a shared database.
LANSA Products	**Integration using Composite Applications**
LANSA Open for .NET	Build applications in MS Office SharePoint Server with components built using C#.NET and programs running on an IBM i server.
Visual LANSA Framework	Assemble composite applications from components built with Visual LANSA and C#.NET.
LANSA Products	**Integration using Portals**
LANSA Integrator	Display customer order status in MS Office SharePoint Server using C#.NET and use LANSA Integrator to marshal the Web services needed to extract the order details from an IBM i server.
LANSA Open for .NET	Build a customer maintenance form in MS Office SharePoint Server using C#.NET and use LANSA Open for .NET to interact with the DB2 database and applications on an IBM i server.
LANSA Products	**Integrating Applications**
LANSA Integrator	Provide a set of services to support integration from an application to an ESB, or point-to-point application integration.
LANSA Open for .NET	C#.NET applications integrated with databases and operating system services running on an IBM i server.
RAMP or Visual LANSA Framework	Include components built with LANSA, C#.NET, RPG and COBOL.
LANSA Products	**Process Integration**
LANSA Composer	Business process orchestration and end-to-end management, calling applications to action the steps in the process.
LANSA Integrator	Participates with the steps in the business process. For example, call a Web service to collect data for a step in the process.
Visual LANSA Framework	Visual LANSA Framework built applications include process integration when calling other components.