Chapter 7

Integration Myths

Drawing on our practical operational experience in application integration projects, we have been able to inventory a well stocked catalog of conventional wisdom on the subject.

The preoccupations that underlie these ideas are generally legitimate. But sometimes – often – the responses to them are “cookie cutter” responses, and can lead to more or less camouflaged failures. This is why it seems important to us to shed some light on them.

7.1. The mirage of the single tool

How many times have we heard or read the following statement: “one single tool responds to all requirements”!

When an IT director (often the Chief Information Officer) takes a long, hard look at the technical architecture of his or her IT department with the goal of technical component evolution, it is not uncommon that a consulting firm, generally after careful study of the different technologies on the market, selects a single technology to treat the entire problem addressed.

The choice is performed by favoring:

– *innovative* technology – after all, architectural consulting firms are also paid for integrating innovation inside IT – otherwise, what use are they?

– *or perhaps just the technologies that the architects are aware of*. What good does it do to change, since the architects in question are fully conversant with the technology? At least they will be fairly certain that their competence will still be required when the technology of today is also the technology of tomorrow.

With regards to application integration, this type of behavior can bring about the failure of the integration solution.

Some examples given below will illustrate the idea.

7.1.1. ***A conservative choice: example and consequences***

For several years, the enterprise ENERGY ONE, positioned in the corporate services market, has been using an ETL tool to input its data warehouse repository. Data is extracted regularly (once per day is functionally sufficient) from production databases, validated and cleaned (with a check on functional doubles), then injected into databases of the data warehouse repository. With the success of this technology in mind, ENERGY ONE decided to generalize the use of the ETL to cover the whole of the application integration problem, specifically for:

– managing intra-enterprise business flows (real-time and “batch file”);

– exchanging with its partners.

As one result, a generalized use of an ETL tool pushes the limits of a commercialized product that was never designed:

– to manage exchanges with a frequency greater than one session per hour. This is normal; this was not its initial objective;

– to pilot file flows with partners at an adequate level of quality of service. The use of FTP upstream and downstream from the ETL does not always allow the integrity of those files to be ensured;

– to verify the quality of services supplied to or received from the partners. The ETL tool provides no function for dataflow supervision;

– to offer real-time exchanges to the departmental applications which had signaled their need for them. The ETL tool functioned only by extracting/integrating files that use databases.

Faced with the lack of enthusiasm raised by the large-scale deployment of this technology, ENERGY ONE found that it had to:

– refocus the use of the ETL tool around its standard function of updating the data warehouse;

– restore the choices around application integration back to the different business departments – the CIO no longer had the credibility necessary to put a solution together.

On balance, the feeling inside the enterprise about application integration was negative. ENERGY ONE will not be restructuring its integration strategy anytime soon. The *status quo* and the principle of "every man (or every department) for himself" will be engraved there for a long time to come.

7.1.2. *"Modern" architectural choice: example and consequences*

Refer here to the concrete example presented in section 11.2.

7.2. XML: miracle format

Another statement we have often read or heard: "all formats must be XML formats – this guarantees the use of standards"!

Taking nothing away from the relevance of XML-type format (see sections 3.2.1.1 and 4.3.2), it is however important to recognize that its success among analysts, architects, software providers, and also among users, has resulted in the construction of a kind of Tower of Babel, perhaps with the same tools, but with bricks of completely different shapes, sizes and colors.

Several hundred and perhaps thousands of business "standards" use the XML format, though no general standardization has yet been defined by a "United Norms" organization, or has been publicly accepted by all stakeholders, even on attributes as universal as names and addresses, for example. A certain number of endeavors are however underway, for example, at OASIS around UBL (see section 4.3.2).

However, in the view of many decision-makers responsible for integration choices, XML offers a certain guarantee of simplification. If this is true for computing applications (which can indeed capitalize on a single grammar for writing or reading the dataflows for which they are responsible), it is inexact if not utterly false in everything that concerns the supposed uniqueness of the formats produced.

There has never been a greater need to for integration engines to transform XML formats to *other* XML formats.

Why should this be a problem? Is this not precisely what integration engines are for?

This is a problem because these transformations have a high cost in terms of performance (see section 3.2.1.1).

Like variable type formats – of which they are only a particular type – XML formats require dataflows to be parsed on input to search for the information to be checked or to be transformed in an integration.

As for transforming a format of the fixed length/position type (where, by definition there is no need to "parse" the information, since each type of data is in the same place in the record or the message), it requires on average *five to ten times as much time to process the same business event in an XML-type format as in a fixed format*. That ratio can vary slightly depending on the commercially available tools.

By way of illustration, an integration engine specialized in fixed position/length formats was able to process *100 million events in four hours on a mainframe CPU*. That performance level is simply not currently accessible to XML formats, even after associating functions of parallelism, multithreading, and scalability proper to a significant number of integration engines on the market.

The "all-XML" approach therefore has a cost that will make a lot of people happy... so long as they are among the providers of the hardware and software resources required to process the required volumes!

The concrete example presented in section 11.2 provides an excellent illustration of this problem.

7.3. Business adapters: simplifying the implementation

We often hear or read the peremptory statement: "the more business adapters in the solution, the simpler it is to implement"!

One of the selection criteria admitted by the entire community consists of searching for market solutions that include the largest possible number of thick (business) adapters.

Let us recall that a business or "thick" adapter (see section 3.1.3.3) is a component – placed between the business application and the integration engine – which provides connectivity functions (notably communication protocol) and which guarantees that the format sent or received conforms to the specifications expected by the business application. Information about the wrong or right integration (in the business meaning of the words) is also part of the package.

Common sense leads then to the idea that the more business adapters that are available, the simpler the integration between the applications.

Faced with these good sense arguments, providers of integration solutions step into the breach by presenting in their commercial brochures the largest possible number of thick (business) adapters. One such software provider – since absorbed by a market heavyweight – went so far as to make that its principal sales argument.

Indeed, this criterion is easy to understand and quantify, initially by potential clients, but also by the sales force for a software vendor, where anything that looks like routinizing the sales cycle is manna from heaven.

How does it happen then that in the real world, application integration based on solutions that comprise a large number of business adapters has produced *no significant benefit* compared to a solution which uses technical adapters – *not in delays, not in implementation workload*?

In a significant number of cases, *the deployment objectives for this type of solution were lowered*. Why?

First of all, by definition, there is no business adapter available on the market for specific applications developed for a given client.

In the case where the integration need is strongly centered on existing, often specific applications, the presence of business adapters in software providers' catalogs has therefore strictly no interest.

However, what about the case of integrating commercial software packages, such as solutions for managing customer relations or production, or the accounting and financial modules at the core of ERP (Enterprise Resource Planning)?

As the real-life example below illustrates, disillusion here can be sharp.

7.3.1. *Business adapter: implementation – maintenance – problem*

As part of our consultancy missions, we were called on by the CIO of a large French industrial manufacturing group who had:

– acquired an EAI solution with a good market reputation;

– chosen a commercial ERP to manage its financial back-office;

– purchased the adapter corresponding to the selected ERP and offered by the provider of the EAI solution.

Concerned with verifying the operational side of the said adapter, the CIO's team were given a convincing demonstration in the software provider's offices.

Once the solution was chosen, the project was implemented – and it is at that point that the problems started.

Extracts from our interview with the CIO:

"First of all, we chose the ERP version that conforms to our business needs. In addition, since part of the interest of ERP is that we can specialize it using its capacity to accept our specific additional parameters and business objects, we did not distrust the adapter. We thought that, in spite of adaptations, it would always function."

"However, when the provider of the EAI solution delivered the corresponding business connector to us, we were incapable of connecting it to the ERP as configured. After consultation, the supplier indicated the following precisions to us."

Response of the provider to the CIO:

"This connector is certified on the version X of the corresponding ERP.

"Its native operation with a different version is perhaps technically possible but does not commit us (which was in fact indicated in the contract, but not emphasized in discussion).

"The ERP parameters which your teams have specialized probably mean that you will have to adapt the adapter.

"We [the provider] *could undertake this adaptation, under the conditions described in your service contract.*

"We remind you that this adaptation does not fall under maintenance clauses of the corresponding software packages, but that it could be the subject of a 'services+' contract, whose measures could be communicated to you."

It therefore became necessary to ... "adapt the adapter"!

Rapidly, a meeting was convened between the provider and the integrator responsible for the EAI solution. It became apparent that the adapter had to be modified to serve the needs of the client. The solution provider accepted that the integrator would ensure the necessary modifications, because the provider did not have the local resources to provide this service. Maintenance conditions for the adapter (ensured by the provider or by the integrator) remained fuzzy.

Results assessment for the enterprise

We advised the enterprise to deploy and use technical adapters as much as possible (files, messages, DBMS, etc.). These adapters are by definition more stable than thick (business) adapters.

The enterprise, which had acquired a significant number of business adapters, reduced their use.

7.3.2. *By way of a conclusion on business adapters*

Too many business adapters can ultimately damage the adaptability of the solution, since the cost of their maintenance becomes very steep with respect to the parameter definitions carried out inside the EAI solution.

We end up then with the paradox that searching for a significant number of business adapters during the Request for Proposal (RFP) phase can in fact often delay later solution deployment.

Even so, when business adapters are stable with respect to changes in the EAI solution and in the ERP, then they ensure relevant connectivity for application integration solutions, in particular by ensuring more advanced tracking for updates to events inside the concerned application.

7.4. Java: the proof of a modern solution

"A modern application integration solution must be constructed in Java!"

Without in any way wishing to undercut the interest of Java either as a language or a development platform, we are forced to observe that, in the domain of application integration, the use of Java should be marked "handle with care".

7.4.1. *The real reason for Java*

As with C, C++ or even COBOL, Java is a development language. Its level of abstraction remains that of a third-generation language. Its readability and maintainability are not easy – in any case, not better than if Java was used natively in the specifically developed applications.

However, part of the interest in an integration solution resides in the simple and "auditable" nature of the parameter settings ensured by the integration broker. The language or "mappings" of the broker must be rapidly adaptable around business developments.

Massive use of a language such as Java entails no significant benefit in terms of maintenance between an integration broker and specific interfaces also written in Java. In this case, bypassing the broker in favor of specific development would save the cost of the licenses.

On the other hand, when significant language power is required, then recourse to third-generation languages – including Java – inside the integration broker is useful.

Java or any other third-generation language can then ensure generic functions such as:

– accessing business repositories;

– performing complex calculations;

– proceeding to global checks and generic types on the events that are checked and transformed.

In the real world, one of the conditions for the success of an integration broker in an enterprise requires a balanced proportion between the use of a third-generation language such as Java, and the parameterization of the broker as such, which should be high-level. Ideally, 80% of the functions for checking, transforming, and content-based routing must be processed by mapping functions or with a high-level[18] language. The remaining 20% falls in the domain of a third-generation language such as Java.

7.4.2. *Limitations of an all-Java integration solution*

The interest of Java as a development platform no longer needs to be demonstrated. Its capacities in deployment, its “natural” scalability for executing processes and its “once-only” development make Java one of the most interesting standard platforms on the market.

For all of that, if integration solutions are “all-Java”, then what about applications that execute on other platforms?

Why should the integration needs of a technical universe such as Windows and above all IBM z/OS simply be ignored?[19]

Here again, it can be useful to analyze the real need of the enterprise in terms of the classification of different operating systems, in order to decide whether or not to depend solely on a Java platform.

7.5. Files: the “poor cousins” of application integration

“File dataflows are a side issue in application integration!”

As we saw in Chapter 3, in the majority of cases, application integration relies on an asynchronous exchange pattern. The application sends information to the application integration solution, not waiting for the response before continuing with other work. The information will be distributed to the different partners at an opportune moment.

MOM tools (see section 3.1.2.3) are natural candidates for “supporting” the dataflows that transit an application integration solution.

There is but a single step between considering file exchanges as outside the spectrum of application integration, and seeing it as the “poor cousin” of dataflow

18 On the other hand, nothing stops this high-level language from being constructed on the basis of macro-functions written in Java.

19 Even if today, Java virtual machines exist for z/OS, their connection with native z/OS applications is not yet entirely operational.

messaging. The step is often taken, because the majority of the offerings on the market were designed to process dataflows in message mode and not in files.

However, the studies on the subject all point in the same direction: *in inter- and intra-enterprise exchanges, file dataflows are in the majority, and represent between 65 and 80% of the total*. Not taking them into account ends up taking care of only about one-third of the needs – at best.

Implementing an application integration solution that must process file dataflows is a response to particular problems, that the simple implementation of file adapters upstream and downstream the integration solution does not resolve:

– Is the file a carrier of one or more functional integrities?

– In case of error detected by the application integration solution on one or more records in the file, should the whole file be blocked, or should all or part of it be allowed through?

– In case of error in processing by the sending application, how is it possible to be sure that sending the application did not re-send the file, including all the initial records that were not in error? And in that case, how is it possible to be certain that the same records are not processed twice?

– The volume of files to be processed is generally much more voluminous than messages. How then can we be certain that the integration engine was designed to handle such volumes?

Certain elements in the response are indicated as part of the concrete example presented in section 11.2.

7.6. Process and services are everything

"Goodbye application integration. Hello processes and services – everywhere!"

As we have previously underlined, the computing industry is perfectly content to stage its own revolutions. A new technology must drive out other, older ones because they are not adapted to the new issues that confront enterprises.

In fact, in the real world, application integration is a good deal more complex and multi-form subject.

It is commonly noted that EAI is mature technology, and that Business Process Management and SOA should now be installed everywhere "in double-quick time". This type of profession of faith has the advantage of simplifying the discourse from suppliers, making sure that potential clients can hear and understand it. It has only two drawbacks: it is dangerous and costly.

It is dangerous to lead business requirement teams to believe that just modeling a process will easily lead to aligning the information system with the business of the enterprise (see section 4.4). If the existing IT applications are not designed with services in mind from the outset, it will be difficult, perhaps impossible, to adapt them. Consequently, new SOAs will have to be developed from scratch, with all the associated delays and costs. It will not make matters better to use the same "design" workstation for setting the parameters in the application integration solutions and modeling the business processes. Indeed, the pace of modification to a business process is different from that of an integration layer. In addition, neither the user populations nor the preoccupations of the two approaches are the same.

It is dangerous to spread the idea that all the business processes in the enterprise must be orchestrated. For example, in an energy company, what is the use of handling the purchase of office supplies in process mode? What business benefits exist when the simple implementation of a B2B platform for management of the orders is quite sufficient?

It is costly to launch a broad-spectrum approach to modeling the processes of the enterprise, without questioning the link between this modeling and the underlying functional and technical models for the applications in the corresponding information system.

As was expressed recently by an architect responsible for large-scale deployment of processes inside his service delivery enterprise: "We underestimated the problems of aligning the business models with the execution models in the information system", a coyly roundabout way of indicating that at the end of four years, only five operational processes had been deployed, and the modeling of 150 others had been committed, after spending hundreds of person-years on the entire project.

We want to be quite clear: we are a long way from the idea of hollowing out the genuine interest in making BPM and SOA solutions available. Still, they must be used advisedly, i.e., on processes and service approaches with high added value that "deserve" spending the necessary means.

7.6.1. *BPM and SOA: top-down approach – from business to IT*

In this approach, processes that create value in the enterprise must be identified to make sure that they are eligible for process industrialization. Then the adaptations required in the information system must be committed to. This top-down approach proceeds from business to IT.

Organizationally in this case, we would expect to find business managers in the front line. They are they ones who necessarily must impose constraints in adapting existing applications that are typically implemented by the project execution teams.

7.6.2. *EAI and B2B: bottom-up approach – from IT to business*

The other processes of the enterprise will not be treated as such but implemented in a context of exchange platforms and/or of the flow manager[20] that implements services for:

– securing the transport layers for the information;

– ensuring functions of information distribution;

– offering services for transformation, routing and control of dataflows.

Organizationally, it is the project execution teams that deploy the required exchange platforms or dataflow managers. The business managers are generally called on to ensure the transformation functions and the business checks.

7.6.3. *Complementary approaches*

Far from being contradictory, the two approaches are in fact complementary. Each approach supplies supervision services that are pertinent for the enterprise:

– technical supervision of the transport layers;

– supervision of the dataflows exchanged in A2A or B2B;

– supervision and audit of technical and business transformations;

– process supervision;

– quality of service supervision;

– business supervision.

20 Or again, the exchange process sub-level.