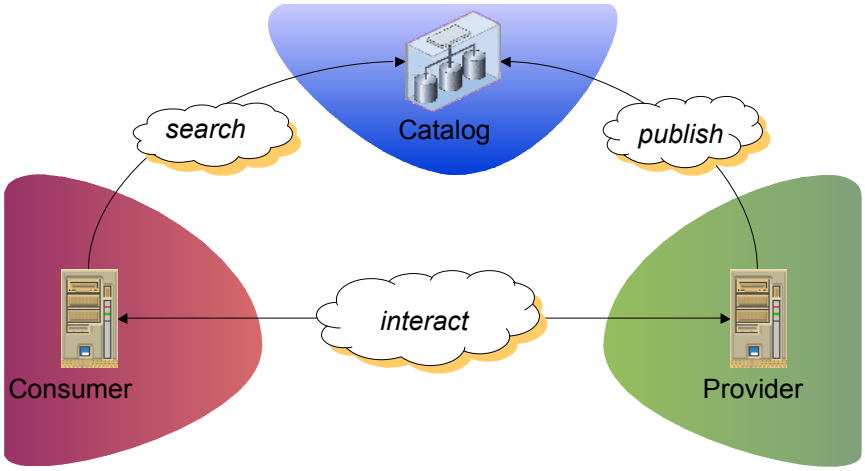
SOA and Gouvernance

Th.Moineau – Thierry@lesMoineau.fr

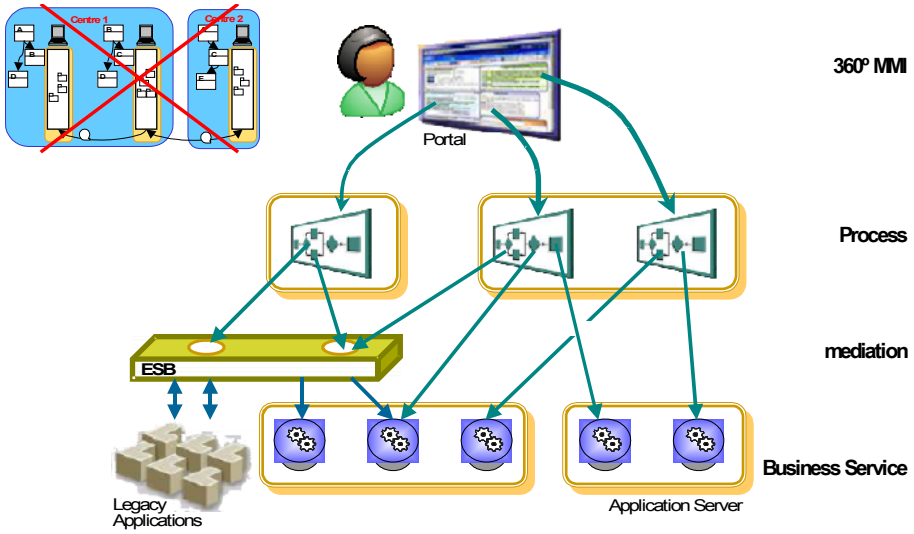
What is SOA ?

There is, today, no standard definition of Service Oriented Architecture (SOA).



From a technical point of view, SOA is a set of technologies coming from the web world and enabling invocation of IT services over Internet : the famous Web Services. More precisely, web services are implemented by a provider, are deployed on some machines and their availability is published in a catalogue. A consumer can then find them in the catalog and use their description to start interacting with the Provider.

So far nothing new (computer have been communicating for a long time – aren't they ?). The revolution is that the interactions are based on standard and simple technologies: XML messages and SOAP protocol over a network, which can be the Internet. This provides the ability of interactions between computers using different technologies (e.g. Java vs .Net) possibly from different enterprises – all of that without the usually long and costly technical integration testing phase.



From a larger Information System point of view, SOA is an enterprise architecture paradigm promising significant reduction of IT maintenance costs.

It aims at moving away from the previous silo-based IT systems (business processes delivered through custom applications running on separate sets of computers and data bases) and their well-known issues of code redundancies, data duplication and inconsistencies.

SOA promotes a 3-layers architecture: a layer of business services, a layer where the business processes are implemented and a man-machine interface.
The business services encapsulate the business data and their fundamental behavior. They are meant to be used by all the business processes, without discrimination, in order to avoid the "silo-effect".
The man-machine interface is not anymore structured by applications but is based on business activities and processes, giving the user a 360° view on the information relevant to his/her work.

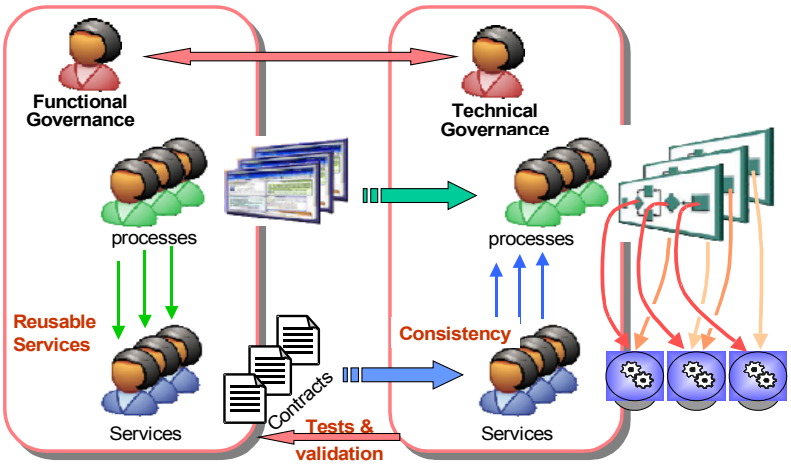
SOA is not technology (not only …)

This looks nice and good and … very technology oriented.

The good/bad news is that, although there are significant technology issues behind SOA, the main problems today are: methodology and governance.

That's what we learned from several large IT system refactoring programs based on Entreprise Architecture and SOA paradigms (including the Copernic program, which aims at a complete refactoring of the entire fiscal IT system : 9 years duration, more than 1000 people within 70 projects and a budget of 900 million Euros - this is probably to date the largest IT system fully implemented according to the SOA paradigm).

Need for transversal Governance structure



Better comfort, efficiency and accuracy for the end users, easier maintenance and lower IT costs : this looks like a target that everybody in any company should be happy to seek for.

From our experience, this is not the case.

First, the business processes are defined by the various business units within the company and this makes sense, as these business units are the owners of the processes. There is however no reasons that business units working independently come up with common reusable services. Indeed, as stated by the French poet, what the processes need are "not utterly different each time, not utterly the same". Without a team responsible to identify these reusable services, they will never exist (identifying the reusable business services is a very complex task, which would request an separate article later). Regarding the specification of the reusable business services, we found out that relying only on a central team is not a good solution. Indeed the amount of detailed business knowledge needed for that would require a huge team and this team rapidly becomes a bottleneck. As most services, albeit reusable across several business units, are naturally closer to one of the business unit, our solution is to have the central team to specify the few services really central and to distribute the other services across the business units.

Then, this central team has to convince the business unit to use these business services ; this means that they have to re-design their business processes according to the reusable business services. Now let's face the real world, the business units usually have very few incentives to do so. Indeed changing a business process is costly for the business unit and if it does create some economy, this is often at company level only and not at the business unit level. Hence this central team must have a very active support from a very high level within the company management.

This is what we call *functional governance.*

At technical level, we envisioned two management structure : (a) a central team implements all the reusable business services and (b) the services are distributed among the business units. In both case, the business processes are implemented by the business units.

The second solution proved more efficient, but requires strong technical governance. Otherwise each business unit would use its own paradigm - for instance, some use RPC style interactions while others use document style interactions, some use session-less transactions while others use stateful transactions, etc. The developers of the business processes would then be faced with a diversity of approach to understand and master. Our experience is that they get rapidly confused, which leads to both a significant number of bugs and potentially to a reject of the approach.

Hence, we recommend that a central team defines and develop a set of architectural rules and guidelines together with technical prescriptions.

This is what we call *technical governance.*

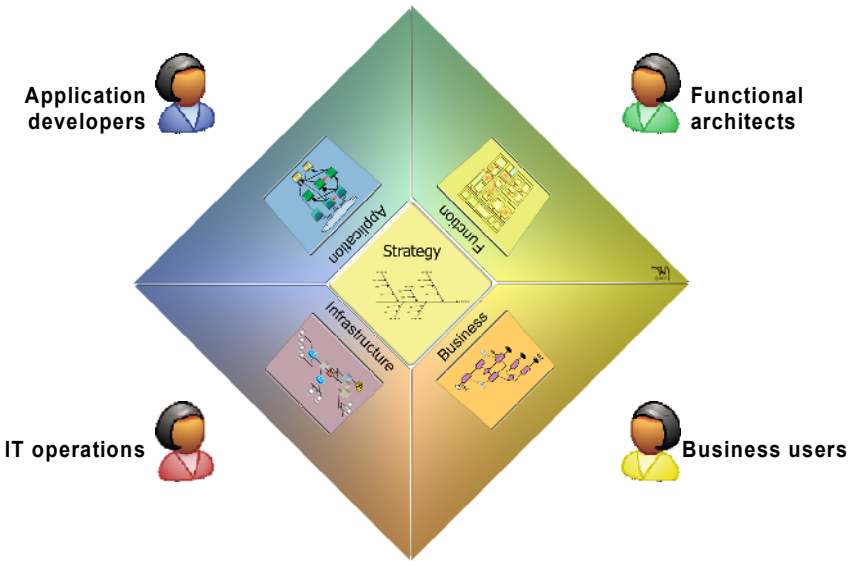
The next issue is related to integration and validation of the reusable business services implementation. The idea, which comes first to mind, is to have the team who specified the service to validate the implementation. This is not perfect as this requires building a complete testing environment, which can be very complex and expensive. On the other hand, one can ask the clients of the services to validate it, i.e. the developers of the business process. This is not perfect either as it easily leads to duplication of tests and to delaying the validation (because not all business processes arrive at the same time). Our recommendation is to have a mix : some basic testing and validation by the specification

team and more thorough testing by a few selected business process developers. The appropriate selection of the tester is a key success factor.

The five views of the Information System

One of the key question we had to face in the Copernic program is "who does what ?".

Indeed, moving from a vertical silo-based architecture to a layered service oriented architecture proved to be a complete mental shift for most participants of the program. If not taken care of, it quickly leads to confusion and potentially blockage of the program (when you ear end-users arguing about the format of the WSDL, then you know that you'll soon be in trouble).



To avoid this confusion, we have used and adapted a method to describe the information system according to five views (which forms a cube – the sixth face is used to pose it on one's desk).

The first view is the Strategy view. This view describes the strategic objectives of the enterprise in term of business and in term of its information system. All the decisions made in the other views must be related to one of the objectives in the strategy view.

The Business view describes the business processes and business activities, independently from the internal structure of information system. This is the view in which live the end users.

The Functional view aims at structuring the information system into processes and business services. This view, which is not meant to be seen by end users, is essential for the efficiency and the maintainability of the information system. This view should be as independent as possible from the technology and technical aspects of the IT system. This is the view in which live the functional architects.

The Application view aims at implementing the functional specification into an IT system. This view deals with application modules and technology choices. This is the view in which live the technical architects and the application developers.

The infrastructure view contains the description of the IT physical infrastructure of the company (computers, networks, …). This is the view in which live the IT operations.

We found out that it is important to avoid hierarchical relationships between the objects in the views, because such relationships tend either to recreate the silo-based architecture or to lead to a confusion of the roles between the actors (e.g. end-users arguing about SOAP encoding rules). Of course consistency rules must be ensured (for instance a machine in the infrastructure view may host several application modules from the application view and vice-versa an application module may be deployed on several machines – but the amount of memory required by all the modules on a machine must be compatible with the physical memory in this machine).

We have defined the interactions between the actors in the different views. We found out that it is critical to close the complete loop through a service level agreement between the business users and the IT operations.

After 2 years of using this gouvernance model, we found this 5 views approach very valuable for the deployment of a SOA-based architecture – definitely more appropriate than the usual layered approaches.