Praxeme's four forceful thoughts for SOA

Purpose

Praxeme is an enterprise methodology. Among others, it contains the necessary processes for the service-oriented architecture and design.

This article summarizes the main messages of Praxeme to make a success of the SOA projects.

Contents

Author Dominique VAUQUIER

Translator Charles SARA

Version 1.1, 23 February 2008

SOA: an IS architecture style

Service, the atom of the information system

Technology provides us with structuring and communication mechanisms which enable to add sense to the metaphor of service, as applied to computer systems. For that is what it is all about: a metaphor. In accordance with its meaning in the acronym 'SOA', s*ervice-oriented architecture*, service is a unit of composition of a computer system. Having regard to this metaphor, the notion of service moreover refers to the smallest processing unit that can be requested. It is therefore an operation rather than a set of operations. You need a piece of information, you set off an action against the system, or you ask for a transformation, etc.: all this is achieved by activating a service inside the system, more or less straightforwardly. In this view, data is masked, protected by services which guarantee the integrity of the system.

Structuring of the system

Computer systems are made up of thousands of such services. Thus comes the issue of grouping them and optimising their structuring. This is where the following concerns start arising: reusability, non-redundancy, loose coupling and volume control, complexity and costs. In a word, we are tackling IS architecture.

Technical architecture

Let us get things straight with the term "architecture". Technical architecture provides us with the conditions of feasibility of services architectures. It can involve new solutions as well as it can include more conventional technologies: the important thing is still the service metaphor. The options of system structuring do not fall within the competence of technical architecture. They fall within the scope of another discipline: logical architecture.

Logical architecture

Once the conditions of technical feasibility settled, SOA is a matter of logical architecture. The question is: "How to structure the computer system at best?". The procedure for it, which is logical architecture, consists in setting up some precepts and topological constraints, and in enforcing some formal rules in order to organise the substance of the system. The logical aspect is only one aspect of the "enterprise system", just in between the "upper-level" aspects (business, organisation) and the computer aspect. The logical model describes the computer system in terms that are quite independent from technologies. This results in two benefits: first, the model is more easily communicable; then, since this model is kept protected against technical evolutions, it offers the stability necessary for a long-term undertaking.

The SOA style

In the course of the history of IT, logical architecture has gone through several periods and styles. The latest one was functional architecture, i.e. system structuring based upon functions or functionalities. In this approach, the prevailing decomposition criterion is the functional domain. In the SOA style, the elementary term is the service, and the issue of the decomposition criterion finds more sophisticated answers.

Cosmetic SOA *versus* overhaul SOA

"Cosmetic SOA"

In publications, the term SOA embraces two practices which differ a lot, both in their purpose and in their repercussions on the computer system. The implementation of a few services over a legacy system allows to make its access easier and to open it up to other systems. It brings a major contribution, but it does not basically change the system itself. We call this practice "cosmetic SOA". Many SOA projects belong to this category.

"Overhaul SOA"

SOA also depicts a more radical approach of the system, an approach which gives to the term "architecture" its full sense. That is what we call "overhaul SOA". This phrase is characteristic of an architectural thinking which aims at restructuring the system taken as a whole, at banishing any single redundancy, and at promoting reusability to a maximum. In this view, the system is no more a set of applications necessarily redundant, but becomes an Erector set of services inside which the same behaviour or rule is programmed once only.

A gradual transformation of the IS

This rebuilding of the system cannot fortunately be achieved with a single step. Indeed, we must admit our joint inability to manage such projects. The benefit of SOA lies in its ability to overhaul the system by starting with a portion of it, progressively. Thus, the projects that produce the services can be kept to a controllable size. Obviously, in order that the system overhaul goes on without veering off course, the IT Departments must comply with several conditions, among which the pre-existence of a rigorously designed target, the homogeneity of the method and a couple of organisational arrangements. It is important to measure the impact on the organisation. Indeed, as the SOA approach subordinates the immediate purpose to the general interest, it ushers the IT Department in a new dynamics and a new relationship between projects and cross activities.

SOA and IS city planning

Differences

In connection with that, we have to mention the relationship between SOA and IS city planning or Enterprise Architecture. At first sight, this relationship doesn't seem obvious, whatever marketing bodge-ups try to pretend. Indeed, the defenders of each of these approaches have a totally different and hardly reconcilable opinion of the substance of the information system.

How does classical city planning consider the IS

The orthodox discourse of IS city planning is characterised by:

- the primacy granted to activity (exclusivity of process modelling, precedence of functional domains);
- the application unit as base component of the computer system.

The IS considered from the standpoint of the SOA trend

In contrast, services architecture – at least in the Praxeme methodology – makes room for constituents which follow from a semantic model, above processes. It introduces "domains of objects" that shelter highly reusable services. "Business" objects don't come second, relative to processes and activities: they come earlier and they have an influence on choices that are very structuring at logical architecture time. Besides, this philosophy of the computer system puts into perspective the application seen as a unit: even though it does not disappear, the application loses its substance and becomes a place for assembling shareable services. These services are designed with a reusability purpose and having in view a comprehensive reorganisation of the system as such. The prevailing viewpoint is not that of the projects: it assumes a higher-level outlook, carried by the organisation of the IT department.

Convergences

Despite these fundamental[1] differences, SOA and IS city planning can converge and reinforce each other. For that purpose, it is helpful to revisit the discourse and practice of city planning, and to adopt a new discourse that fits into a comprehensive methodology. The convergence between SOA and IS city planning comes to light as we get back to their aim. There is indeed a common purpose : both approaches aim at having the system evolve towards an optimal structure. Their positioning is or should be the same : wide scope – system scale, enterprise consideration – and long term. This posture is natural and, one could say, constitutive of IS city planning as it was first formulated by Jacques Sassoon. It is yet less natural on the SOA side, since this approach mainly originates from technology.

Divergences

When we compare the representations produced, on the one hand, by classical town planners and, on the other hand, by logical architects, we notice that the city planning targets, zoning regulations and other land use plannings, merely come down to applying divisions to the system. These representations are static and exclusively functional[2].Instead of helping, the addition of "repositories" and "data deposits" reveals that this approach doesn't comply with the principle of encapsulation. According to this principle, the data is not visible from the outside: we can have access to it only through… a service. This same principle of encapsulation lies in the very heart of the concept of service. According to the SOA style, the logical model consists in having a sphere of services dominate and conceal a sphere of data.

[1] I am aware of having slightly overdone it. Shall I say, caricature is an educational means. *For the sake of argument …*

[2] The term "functional" must be here understood in its strict sense : related to the function, thus connected with action, activity. To us, functional architecture is a logical architecture whose structuring criterion is the function. I wouldn't mask my perplexity concerning approaches which conciliate a functional level and a logical level. Such a verbose framework generates useless works.

The logical architecture graph

The logical architect, endorsing the SOA style, develops the city planning target in the form of a logical architecture graph. At first sight, it corresponds to the same representation as the one drawn up by the city planner. Yet, the art of the logical architect is much different: he adopts a stricter notation which enables him to study the dynamics of the system at the same as its statics. As he splits up the system into large constituents, he immediately cares about the dependencies implied by this division, about the generated calls and exchanges. He would cut here rather than there because he knows that this option optimises the system from the angle of its behaviour. Besides, the logical architecture graph is just the first among the diagrams that reveal the logical model. With the same notation, the logical designer will carry on with his work down to the detailed specification of logical services. Thus, there is a continuity of the activity line, and the developer will exploit the logical model without trouble: he will translate it into software, by applying the technical architecture choices. This continuity principle removes the blame put on city planners by developers: producing a large map which is irrelevant to projects.

The role of the IS city planner

Therefore, what is left to the IS city planner, which could not be tackled by the logical architect? It is his part as a ferryman between business and IT. The IS city planner is the only one able to collect the enterprise strategic orientations, to care about the general business concerns, to anticipate business evolutions and to sense their impacts on the IT tool. This ferryman part cannot be tackled in earnest by the logical architect : he is too much worried about the computer system itself, at the risk of yielding to the attraction of formal considerations, far away from the business.

To conclude this paragraph, the IS city planner and the logical architect are complementary and need to collaborate with each other, since both disciplines deal with the same aspect: the logical aspect. This aspect is deliberately appointed as an intermediary between the business and the IT.

SOA: the method

Questions

The implementation of a services architecture raises many practical questions, especially the following:

- How to determine the services?
- How to structure the system?

In the absence of answers to the first question, projects might only produce accessors, services like: create, update, delete. Is this not much ado about nothing? Even if we avoid such extremes[3], the lack of granted answer causes the heterogeneity of practices, between multiple designers, between multiple projects… This situation ruins any hope of improving the system.

The issue of right structuring must also be accurately answered.

Answers

It is the job of methodology to answer such questions and to explain what to do. The Praxeme open method addresses both questions, among others. It is based upon a reference framework which does not limit to the logical aspect, but embraces all of the aspects of the "enterprise system". This covering represents one of the necessary conditions for carrying out such activities as logical architecture and IS city planning. Indeed, Praxeme prescribes a procedure for designing services by deriving upstream models. Praxeme lays down two models as prerequisites to the logical model:

- the semantic model, which expresses the basics of the business,
- the pragmatic model, which describes the enterprise activity (organisation, processes, use-cases).

Upstream models

In order to be exploitable, these both models comply with definite criteria. For instance, the semantic model does not limit to what we too often call a "business objects model": it is much more than a bad conceptual model of data. Similarly, in order to achieve the derivation of a pragmatic model, we cannot just consider use case diagrams as they can be found with most of the projects: they mask a too large redundancy.

The logical aspect disciplines

About the logical aspect, Praxeme makes the difference between the disciplines of IS city planning, of logical architecture and of logical design. These activities hang together carefully. The use of a standard notation, in this case, UML, makes the works much easier. Another vital standard for this IS production line is the OMG standard: MDA (model driven architecture). The technique of transformation of models helps us reach the goal: we are now able to set up software industrialisation. Praxeme provides a crucial step called "logical/technical negotiation", which is the place where logical and technical architects make sure the logical expression can be converted into software. A lot of derivation rules between models are mechanical and can get automated.

[3] Extremes, but no caricature : this is a frequent observation

Conclusion

In order to get full benefit from the SOA architecture style, it is necessary to re-examine from top to bottom the activity line which goes from objectives and business down to distribution. Technical architecture fixes the conditions of feasibility of the services architecture. Logical architecture develops the structuring of the computer system by reviving functional domains, but also draws its inspiration from domains of objects, found in the semantic model.

Mastering these levels of concern requires a comprehensive method. Praxeme provides such a method, along with the practical and economical advantage of open source: a reference method, widely shared and of easy access. The example of the large project at SMABTP is a perfect demonstration of the contribution of the method, when it is supported by the management.

For further information, and in order to get a copy of the methodological guides: http://www.praxeme.org

Dominique VAUQUIER - mailto:dvau@praxeme.org