Praxeme's four key concepts for SOA

Objective

Praxeme is an enterprise methodology. It contains, notably, the procedures needed for designing service-oriented architecture.

This article summarizes Praxeme's main messages for carrying out SOA projects successfully.

Contents

- SOA: an IS architectural style
- Cosmetic SOA *versus* overhaul SOA
- SOA and IT city planning
- SOA:

Author Dominique VAUQUIER

Translator Joanne TOWARD

Version 2.0, 1 May 2015 (first version: 7th April, 2007)

SOA: an IS architectural style

Service, atom of the information system

Technology provides us with mechanisms to structure and to communicate. This allows us to give meaning to the metaphor of service, as applied to information systems, for it is a metaphor that we are talking about. In the abbreviation SOA (service-oriented architecture), "service" is taken to mean a unit of composition in an information system. If we respect this metaphor, the notion of service refers, moreover, to the smallest processing unit that can be requested. It is therefore an operation rather than a set of operations. For example, I need a piece of information or I initiate an action on the system or I request a transformation, etc. This is achieved by activating a "service" in the software system, in a more or less direct manner. In this view, the data is masked, protected by services that guarantee the integrity of the system.

The structuring of the system

The information system is made up of thousands of such services. This then raises the question of their grouping and their optimal structuring, which is where concerns about reuse, non-redundancy, loose coupling, volume control, complexity and costs come in. In short, we are tackling IS architecture.

Technical architecture

We have to agree on the term "architecture". *Technical architecture* provides the feasibility conditions for a service-oriented architecture. It can turn to new solutions but, equally, use more conventional technologies: what remains important is the metaphor of service. Choices about how the system is structured do not fall within the remit of technical architecture. They involve another discipline: *logical architecture*.

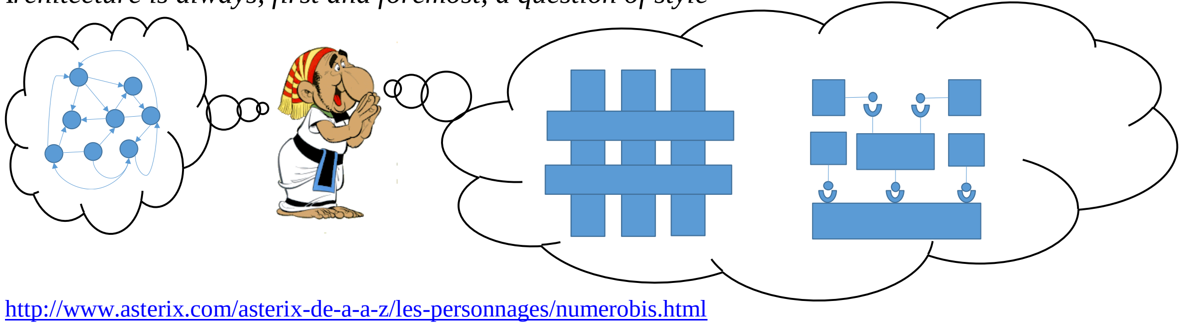
Logical architecture

Once the technical feasibility conditions have been established, SOA is a matter for logical architecture. The question is: "how can we best structure the information system?" The procedure consists in applying some precepts and topological constraints, in imposing several formal rules, in order to organize the substance of the system. The logical aspect is an aspect of the "enterprise system", an intermediary between the "upstream" aspects (business, organization) and the IT aspect. The logical model describes the information system in terms that are relatively technology independent. Two advantages stem from this: first, the model is more easily communicable; second, it benefits from the stability required for a long-term enterprise as it is protected from technical evolutions.

The SOA style

Throughout the history of IT, there have been several periods and several styles of logical architecture. The latest one was functional architecture: system structuring based on 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 answers to the question of the decomposition criterion are more sophisticated.

Figure SLB-15_1. Architecture is always, first and foremost, a question of style



© Goscinny & Uderzo, http://www.asterix.com/asterix-de-a-a-z/les-personnages/numerobis.html

Cosmetic SOA *versus* overhaul SOA

"Cosmetic SOA"

In publications, SOA covers two very different practices, both in their purpose and in the repercussions on the information system. Implementing several services on a legacy system allows us to facilitate its access and to open it up to other systems. It is a significant contribution but it does not fundamentally modify the system itself. We call this practice "cosmetic SOA". Many SOA projects fall into this category.

"Overhaul SOA"

SOA also describes a more radical approach of the system, one that gives full meaning to the term "architecture". This is what we call "overhaul SOA". This phrase is characteristic of an architectural thinking that aims to restructure the system, seen as a whole, banishing all redundancies and pushing reuse to its limits. In this view, the system is no longer a set of applications, one which is obviously redundant, but an "Erector set" of services in which the same behavior or the same rule are only programmed once.

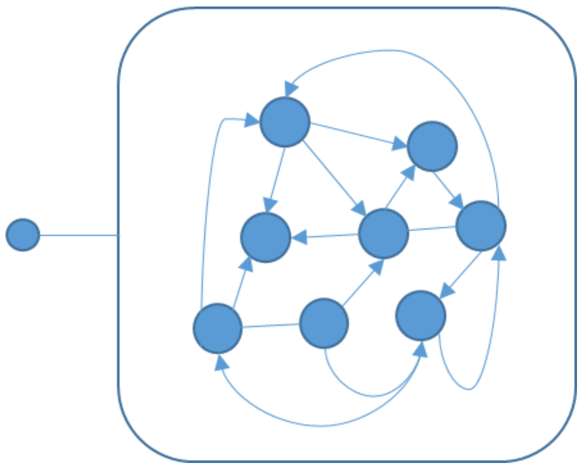
A gradual reform of IS

This rebuilding of the system is thankfully not done in one go. We have to admit our collective inability to manage such projects. The advantage with SOA lies in the possibility of gradually overhauling the system, starting with one part. Thus, service-creating projects remain at a manageable scale. Obviously, for the system overhaul to move forwards without deviating from its trajectory, IT departments must respect several conditions, among them the pre-existence of a rigorously designed target, the homogeneity of the method and a few organizational arrangements. It is important to measure the impact on the organization. Indeed, as the SOA approach subordinates the immediate purpose to the general interest, IT departments are being ushered into a new dynamic, with a new relationship between projects and transversal activities.

Figure SLB-15_2. System quality according to the degree of SOA

Cosmetic SOA

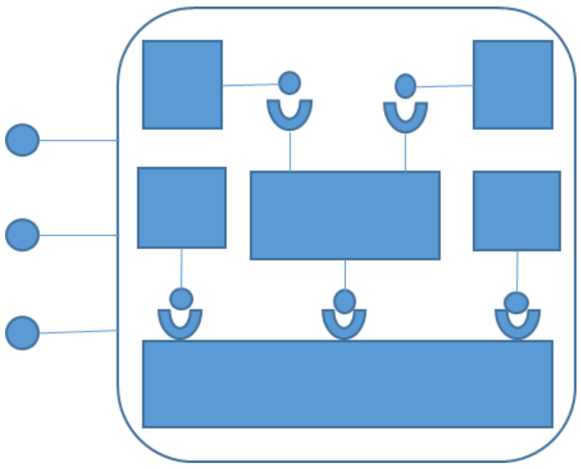
The system provides several services but its internal structure has not changed.



Reuse and internal quality have not progressed.

Overhaul SOA

The system provides all the more services (interoperability) as it has been correctly structured.



The system is itself an assembly of services, with redundancy and controlled coupling.

SOA and IT city planning

The differences

This is an appropriate time for us to mention the relationship between SOA and IT city planning.

At first glance, this relationship is not a spontaneous one, no matter what claims marketing may cobble together. Indeed, the advocates of each approach have a totally different idea, and one that is hard to reconcile, about the substance of the information system.

How classic city planning views IS

The orthodox discourse on IT city planning is characterized by:

- the primacy given to the activity (exclusive use of business process modeling to describe the business, prevalence of functional domains);
- the application unit as the basic component of the information system.

How the SOA trend views IS

In contrast, service-oriented architecture – at least, in the Praxeme method – makes room for constituents that are deduced from a semantic model, upstream of the business processes. It introduces "object domains", housing highly reusable services. The "business" objects are not secondary to the business processes and activities: they come before them and influence the highly structural choices in the logical architecture. Moreover, this philosophy of the information system puts the application as a unit into perspective: without disappearing, the application is emptied of its substance and becomes an assembly point for shareable services. These services are designed with reuse in mind and in the prospect of fully reorganizing the system as such. The main viewpoint is not a project one: it presupposes a higher-level viewpoint, carried by the organization of the IT department.

Convergences

Despite these fundamental differences[1], SOA and IT city planning can converge and mutually strengthen each other. To do so, we should revisit the city planning discourse and practices and adopt a new discourse, integrated within an overall methodology. The convergence between SOA and IT city planning appears if we return to their aims. There is, in fact, a shared intention: both approaches seek to make the system evolve towards an optimal structure. Their positioning is, or should be, the same: global in scope – scale of the system, consideration of the enterprise – and long term. This position is natural and, it can be said, constituent of IT city planning (IS urbanization), as first formulated by Jacques Sassoon. It is less so on the SOA side, as the origin of this approach is essentially a technological one.

Divergences

If we compare the representations made by classic city planners, on one side, with those made by logical architects, we can see that most of the city planning targets (such as "local town planning" and "land use planning"), limit themselves to dividing up the system. These representations are *static* and exclusively *functional*[2]. Far from improving things, the addition of "repositories" and "data deposits" show that this approach has not adopted the principle of encapsulation. According to this principle, data is not seen from the outside: it is only accessed by… a service. This same principle of encapsulation lies hidden at the very heart of the concept of service. In the SOA style, the logical model is such that a "services" plan has a view over, and masks, a data plan.

[1] I am fully aware that this is a slight exaggeration. For the sake of argument, let us say that the caricature has its educational purpose…

[2] The term "functional" is taken, in this paper, in its strictest sense: related to the function, therefore linked to the action or the activity. For us, functional architecture is logical architecture, for which the structuring criterion is the function. I cannot hide that I am left perplexed by approaches that reconcile a functional plan with a logical plan. A framework that is so verbose generates unnecessary work.

Logical architecture graph

The logical architect, adopting the SOA style, draws up the IT city planning target as a logical architecture graph. At first glance, it is the same thing as the representation made by the IT city planner. Nevertheless, the art of the logical architect is a very different one. Logical architects adopt a more rigorous notation, enabling them to study the dynamics and statics of the system at the same time. When they divide the system into large components, they take an interest in the dependencies that this division implies, in the calls and exchanges that it will generate. If they make the division in one place rather than another, it is because they know that this choice optimizes the system's behavior. Moreover, the logical architecture graph is only the first of the diagrams that show the logical model. With the same notation, the logical designer's work continues right up to the detailed specification of the logical services. There is, therefore, continuity in the activity chain and the developer can seamlessly exploit the logical model: it can be translated into software, by applying the technical architecture choices. This principle of continuity silences the criticism aimed at IT city planners, from the developers' perspective: producing a Magna Carta that does not concern the projects.

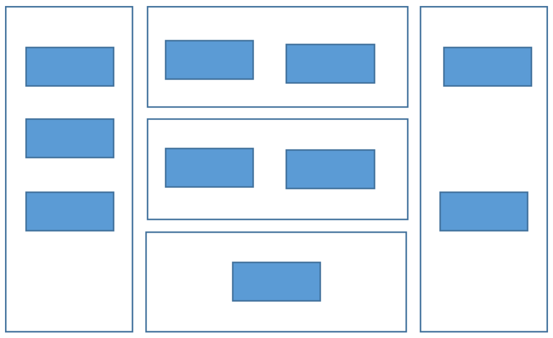
The role of the IT city planner

What is left therefore for the IT city planner to do that could not be done by the logical architect? Well, the role of mediator between business and IT. The IT city planner – or enterprise architect – is the only person able to gather the strategic directions of the enterprise, to take an interest in the general business concerns, to anticipate their changes and sense what impact they might have on the IT tool. The logical architect cannot take on in full this role of mediator: he or she is too preoccupied by the IT system itself, at the risk of being tempted by formal considerations, irrelevant to the business. The logical architect is more on the technical side than a business actor.

To conclude this paragraph, there is a complementarity and need for collaboration between the IT city planner and the logical architect, with both disciplines intervening on the same aspect: the logical aspect. This aspect is deliberately positioned as an intermediary between business and IT. As such, it intervenes in the go-between function between the business and the technical solution, a function taken on by the IT city planner or the enterprise architect. Beyond this go-between function, the technical system has to be fully designed, at the requisite levels of detail that are needed for the system to work and for it to have the required qualities. The logical architect is responsible for this design work.

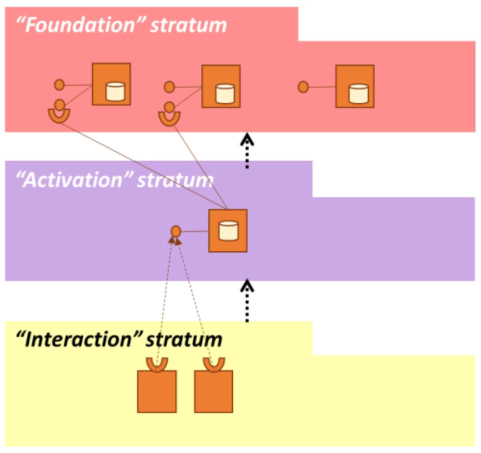
Figure SLB-15_3. Two types of representations of the information system: the "land use planning" and architecture graph

Typical diagram of "land use planning" in IT city planning practices



The representation shows the statics of the system. It does not allow us to study any dependencies created nor to show redundancy nor to anticipate the dynamic behavior of the system.

Typical diagram for an architecture graph according to the Praxeme method



The system architecture obeys strict topological constraints. It is described in such a way that we are able to assess its quality.

SOA: the method

The questions

Implementing a service-oriented architecture raises many practical questions, in particular:

- How do we find the services?
- How do we structure the system?

In the absence of answers to the first question, there is a risk that the projects only produce accessors, services like: create, modify, and delete. Is this not much ado about nothing? Even if we avoid this extreme[3], the lack of answers provided leads to the heterogeneity of practices, between several designers, between several projects… This situation ruins any hope there might have been of improving the system.

The question of the right structure also calls for precise answers.

The answers

The role of methodology is to answer such questions and explain how to do it. The Praxeme open method provides, among other things, answers to both questions. It is based on a reference framework that is not limited to the logical aspect but covers all aspects of the "enterprise system". This coverage constitutes one of the conditions to accurately conduct activities such as logical architecture and IT city planning. Indeed, Praxeme recommends a procedure for designing services, derived from the "business" models. Praxeme sets two models as prerequisites to the logical model:

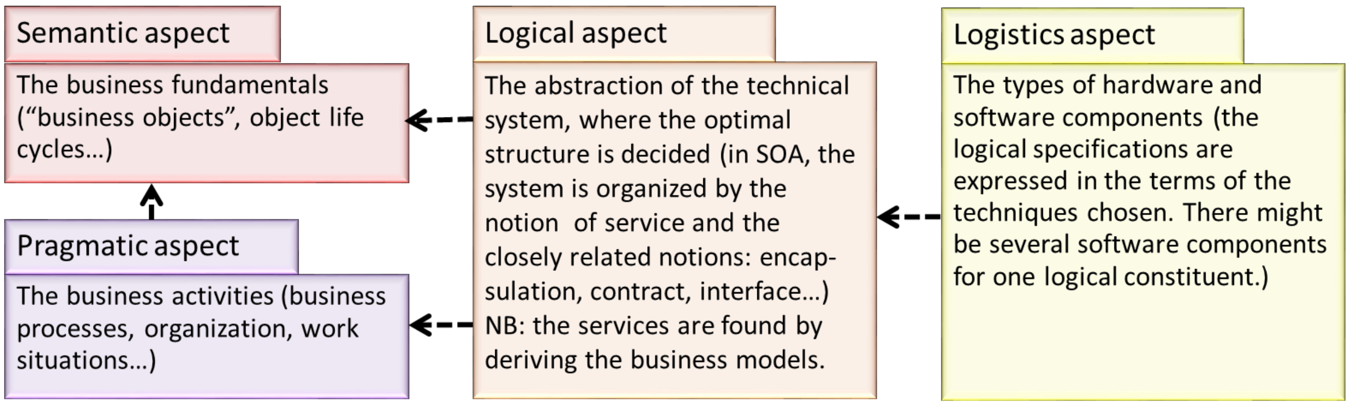
- the semantic model, which expresses the business fundamentals;
- the pragmatic model, which describes the activity of the enterprise (organization, business processes and use cases).

The "business" models

Both of these models, to be exploited upstream, obey precise criteria. For example, the semantic model does not limit itself to a conceptual data model. Likewise, to apply derivation to a pragmatic model, we cannot content ourselves with using use case diagrams, such as can be found in most projects: they are too full of redundancies.

The derivation rules can only be applied on "well-formed" and correctly structured semantic and pragmatic models. These produce, in a mechanical fashion, the services and logical constituents, at least most of them.

Figure SLB-15_4. The positioning of the logical aspect[4]


Semantic aspect
The business fundamentals ("business objects", object life cycles…)
Pragmatic aspect
The business activities (business processes, organization, work situations…)
Logical aspect
The abstraction of the technical system, where the optimal structure is decided (in SOA, the system is organized by the notion of service and the closely related notions: encap-sulation, contract, interface…) NB: the services are found by deriving the business models.
Logistics aspect
The types of hardware and software components (the logical specifications are expressed in the terms of the techniques chosen. There might be several software components for one logical constituent.)

[3] Extreme but not a caricature: it is a frequent observation.

[4] For more details, see "The fundamental sequence" (reference SLB-56).

The logical aspect disciplines

On the logical aspect, Praxeme distinguishes between the disciplines of IT city planning, logical architecture and logical design. These activities are carefully linked together. Turning to a standard notation, in this case UML, greatly eases the work. Another essential standard for this IS production chain is that of MDA (model driven architecture) from the OMG. The transformation technique of the models means that our goal is in sight: we are now able to implement software industrialization.

Praxeme allows for a crucial action called "logical / technical negotiation", when the logical and technical architects ensure that the logical expression can be converted into software. Many of the derivation rules between the models are mechanical and can be automated.

Conclusion

To fully benefit from the service-oriented architecture style, it is important to take a fresh look, from top to bottom, at the activity chain, from the objectives and business to deployment. The technical architecture fixes the feasibility conditions of the service-oriented architecture. The logical architecture works out the structure of the information system by picking up the functional domains once more and also by drawing inspiration from the object domains, found in the semantic model. The result is a radically new and optimized architecture.

Mastering these levels of concerns presupposes a complete method. Praxeme provides such a method, with the practical and economic advantage of being open source: a reference method, widely shared and easily accessible. The case of the large-scale project by SMABTP is a perfect demonstration of the contribution the method can make, when it is championed by management. 50% of the code was generated from an extremely precise logical model where the logical services were specified in a pseudo-language. Right from the end of the first project, the average rate of reuse was 3.

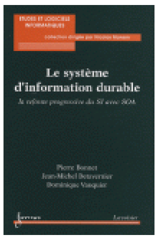
Dominique VAUQUIER - mailto:dvau@praxeme.org

Bibliography

For more information and to obtain the methodological guides: http://www.praxeme.org

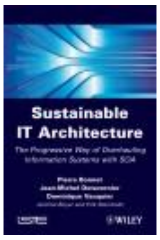
The design and logical architecture procedures of the Praxeme method for SOA are taught in the "SOA, design of a service-oriented architecture" course: (see http://www.praxademia.com/formation/).

The experience of the information system overhaul carried out by SMABTP, using Praxeme, has been described in the following published works:



"*Le Système d'information durable, La refonte progressive du SI avec SOA*", Pierre Bonnet, Jean-Michel Detavernier, Dominique Vauquier, Hermes Lavoisier, 2007

On this large-scale project, Jean-Michel Detavernier was the IS Deputy Director, Pierre Bonnet head of the technical architecture unit, and Dominique Vauquier the logical architect and methodologist. The method gave rise to other applications, in diverse sectors.



"Sustainable IT Architecture, The progressive way of overhauling information systems with SOA", Pierre Bonnet, Jean-Michel Detavernier, Dominique Vauquier, Jérôme Boyer, Erik Steinholtz, Wiley, 2009

Over time, the method has been enriched and consolidated in version 2, disseminated by the company Praxademia, contributor to the open method.

Training

The Praxeme method for SOA is the part of the open method dedicated to the design and architecture procedures in the SOA style.

The “SOA, design of a service architecture” course develops the skills required to identify and specify the services in order to structure them in an optimal manner and to prepare to convert them into software. It also covers the tests, deployment and execution of the services.

The figure opposite shows the content, organized into pedagogic sequences of 30 to 45 minutes, in accordance with the pedagogy-by-objectives method.

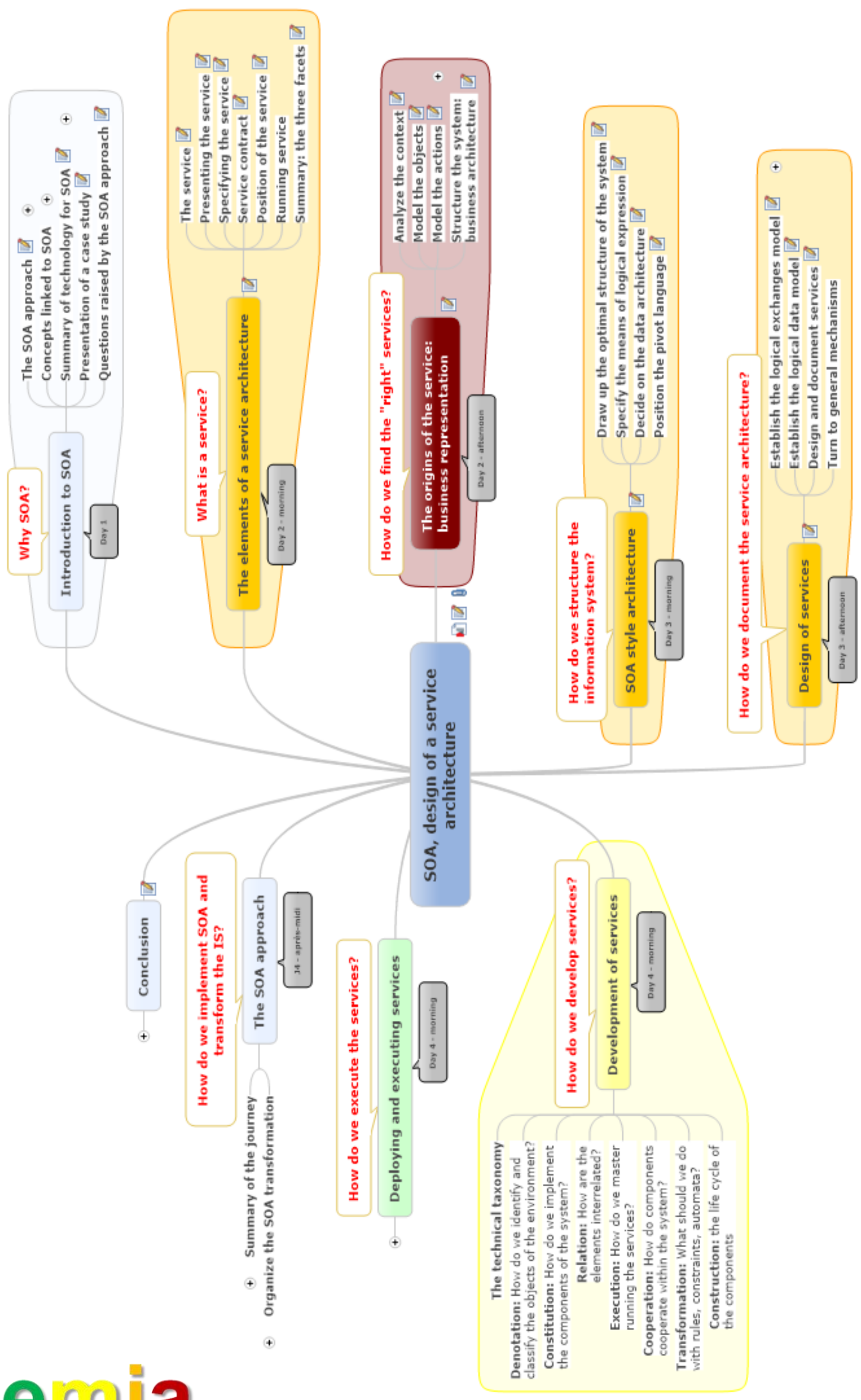

SOA, design of a service architecture
Why SOA?
Introduction to SOA
Day 1
The SOA approach
Concepts linked to SOA
Summary of technology for SOA
Presentation of a case study
Questions raised by the SOA approach
What is a service?
The elements of a service architecture
Day 2 - morning
The service
Presenting the service
Specifying the service
Service contract
Position of the service
Running service
Summary: the three facets
How do we find the "right" services?
The origins of the service: business representation
Day 2 - afternoon
Analyze the context
Model the objects
Model the actions
Structure the system: business architecture
How do we structure the information system?
SOA style architecture
Day 3 - morning
Draw up the optimal structure of the system
Specify the means of logical expression
Decide on the data architecture
Position the pivot language
How do we document the service architecture?
Design of services
Day 3 - afternoon
Establish the logical exchanges model
Establish the logical data model
Design and document services
Turn to general mechanisms
Conclusion
How do we implement SOA and transform the IS?
The SOA approach
J4 - après-midi
Summary of the journey
Organize the SOA transformation
How do we execute the services?
Deploying and executing services
Day 4 - morning
How do we develop services?
Development of services
Day 4 - morning
The technical taxonomy
Denotation: How do we identify and classify the objects of the environment?
Constitution: How do we implement the components of the system?
Relation: How are the elements interrelated?
Execution: How do we master running the services?
Cooperation: How do components cooperate within the system?
Transformation: What should we do with rules, constraints, automata?
Construction: the life cycle of the components

Figure SLB-15_5. Content of the “SOA, design of a service architecture” training course

The full training data sheet is available at: http://www.praxademia.com/formation/soa-conception-et-architecture-dun-systeme-de-services/