

Modeling transportation systems: a case study with the open method Praxeme

Dominique VAUQUIER¹

Abstract This paper is based on the outcome of a research project developed at the RATP Group (the world's fifth largest public transport company). This ongoing research project aims to provide RATP with a proper methodology that will help it to cope with complexity and to address the challenges of the future market (competition, large programs...). The company has chosen the open method Praxeme and was willing to check whether it could apply to the realm of transportation. Praxeme proposes an efficient architecture framework that links the aspects of the Enterprise System together. This framework lays the groundwork for a comprehensive approach to socio-technical systems; it strongly ties the various models together moving progressively from clarified business knowledge to concrete solutions. A very specific case study was used to demonstrate the value of a modeling approach in the eyes of the decision-makers. The approach proved able to oversee issues in the running of the system.

Keywords: transportation, modeling, UML, methodology, Praxeme, system

Introduction

Context

The RATP Group is the world's fifth largest public transport company operating all modes of collective mobility – bus, metro, trains and trams. In Île-de-France it runs, maintains, and develops one of the world's densest multimodal networks. Every day it transports over 10 million people. In addition to its traditional role in and around Paris, RATP and its subsidiaries export this expertise across all continents. The offer relies on transport systems that combine numerous elements developed by many industrial providers. Any failed element may cause the failure of

¹ Praxeme Institute, a not-for-profit association whose purpose is to develop and promote the open method Praxeme. See at www.praxeme.org; dvau@praxeme.org.

the entire system and damage the service, with an impact on the comfort and satisfaction of tens of thousands of clients.

In the face of aging lines and upcoming requests for proposals, the “*Maitrise d’ouvrage des transports*” department, in charge of running the lines, launched a research project to investigate the field of methodology, in quest of a method that would allow it to improve its activity and its relationships with its providers. The challenge is to establish new practices in order to tame the complexity of transportation systems.

Research project

The first step was to explain the principles and to adapt the Praxeme methodology and terminology to the context. Praxeme is an open method that results from an initiative supported by the French administration and many public and private organizations (the French army, SAGEM, CNAF, AXA Group...). It offers a comprehensive approach that covers every aspect of the enterprise. In fact, changes to the method have been minor as the people involved have been willing to stick to the open method as much as possible. We only had to change one dependency in the architecture framework, also referred to as the Enterprise System Topology². Nevertheless, this was a significant change, which marks the adjustment of the methodology to physical systems, in addition to its applications to information systems and human organizations.

Then, a proof of concept was used to confirm the validity of the approach: this will be the case study detailed hereafter. The challenge consisted in demonstrating that, had the method been used, some issues encountered in the current systems could have been anticipated and avoided. At stake are:

- the quality of service,
- cost-cutting thanks to a more rigorous approach to specifications,
- renewing the relationships between the system owners, the engineering department and the industrial providers.

After a brief review of the difficulties encountered, we will set forth the general answers provided by the Praxeme method. We will then look firstly at the findings from a system level and secondly from the detailed case-study level. The case study was chosen by the project sponsors in order to assess the validity and applicability of the method.

² For a presentation of the Enterprise System Topology, see “Enterprise Methodology: an Approach to Multisystems”, in *Complex Systems Design & Management*, 2010 or the white paper and the general guide available on the Praxeme Institute’s website.

Difficulties encountered when addressing transport systems

Complexity or complication

Transport systems are said to be complex systems. This is not something we will dispute. That's right, blame complexity! However, a great deal of the difficulties do not stem from the system itself but from the human beings involved and their lack of rigor. Vocabulary – a loose, ambiguous and unstable terminology – arises as the first obstacle on our path to understanding and designing the system. We were able to collect several glossaries but we found that defining terms was not an easy task.

Moreover, it proved very difficult to build a picture of the systems that were abstract enough from the very details of the current implementations. A lot of functions are named after the solutions provided by the industry. As solutions vary from one line to another and even more from one means of transport to another (metro, bus, tram), building a common and generic representation seemed out of reach, and some participants simply dismissed this target. Even the boundaries of the subsystems did not resist to thorough examination, despite the creed that this exercise is easier when it comes to physical systems as opposed to organizations. Habits and preconceived ideas hinder the reflection and prevent us from seeing the fundamental simplicity behind the accumulated complications.

The curse of requirements

As in most places within the industry sector, the description of the systems is made up of an almost overwhelming number of requirements. This description suffers from the usual flaws:

- The requirements are badly formulated, using an ambiguous vocabulary and relying on assumptions that are not always clarified.
- They are poorly managed, as they exist in the form of documents. A given requirement may appear in several places within the entire documentation and may also refer to several other document elements.
- Taken at the overall level of the transport network, the documentation shows a huge redundancy rate, due to the absence of an overall endeavor to arrange it in a proper structure that would embrace all lines and transport types. Redundancy raises the risk of discrepancy, which, in turn, translates into additional costs and operational risks.

To take an example, ergonomics entails hundreds of requirement elements. Not only are these duplicated but the prescriptions vary from one line to another, in-

cluding items such as the color code or button shapes. This leads to increased training efforts and costs when a driver moves from one line to another; it also multiplies the investment and maintenance of software interfaces.

Where does the truth lay?

In theory, the system owner is the sole authority on the functional specifications which are supposed to perfectly reflect the reality of the system, at least from a functional perspective. In practice, things go slightly differently. Firstly, the functional nature of the specifications is hardly preserved at all, since the physical systems tend to be perceived through their current solutions and terminology rather than in terms of functions. Writing pure functional specifications demands an effort of abstraction and a specific sense of concept over percept. Secondly, even though the system owner develops a first version of the functional specifications and engages his or her own responsibility, the industry provider responds with another document. This document, called a technical specification, partly rephrases the requirements in a more “concrete” way, and partly complements them with additional and technology-oriented requirements. Afterward, the technical specifications will serve as sole reference, replacing the initial functional specifications. The latter do not evolve anyway and are rarely updated because they are seen as less detailed and less accurate than the former. The end result is a functional repository that lacks consistency, accuracy and relevance. Instead, the only valid description of the system – at least in its current state – is the technical one, in the hands of the industrial provider. This says a lot, as far as autonomy and control are concerned.

After a while, the only document which all players, including the system owner, refer to is the technical document. This tends to bias the understanding and definition of the system. Obviously, it hinders new design and slows innovation down, as will be demonstrated in the last part of this paper.

Quantity and quality of the documentation

One can easily imagine the amount of documents that such a transportation company has developed over time. When the sole tool is the word processor and the only unit is the document itself, there can be no way to guarantee consistency and limit redundancy. In reality, we were confronted by several discrepancies and, at some point in the heat of the discussion, the best expert raised an outcry about the regulations: “Hey! The regulations are wrong!” That day, I went back home on foot.

To a certain extent, quantity harms quality, especially when the structure of the corpus has nothing to do with the inner structure of the reality observed. This happens with the traditional approach based on documents or spreadsheets, the very structure of which is detrimental to the natural organization of any system. The mismatch of the description feeds the complication of the design, which augments the complexity of the system.

Initial answers by the Praxeme method

Praxeme presents itself as an enterprise methodology, which means that it aims to cover every aspect of the enterprise, from strategy to deployment. Indeed, Praxeme insists on the need for linking together all specialties and disciplines that contribute to think, design and transform the systems. By “enterprise”, we mean any kind of organized and willful entity or action. By “Enterprise System”, we assert the rational approach to the enterprise and the use of the intellectual tools we find in the system theory³.

Praxeme provides the practitioners (strategists, organization designers, architects, modelers...) with a set of techniques, split over its framework. There are numerous elements to be considered and decisions to be made about the Enterprise System. The framework – the Enterprise System Topology – distributes these elements and decisions throughout well-connected aspects. Here, the notion of aspect is key. It must not be confused with either view or layer⁴. The aspects are supposed to be embedded in the very nature of any system. This tenet – quite metaphorical – sets the stage for the process of analysis and design, at every level and in every dimension, maintaining the natural articulations between the artifacts produced. Thanks to this foundation, Praxeme can easily link the disciplines together into an interdisciplinary approach to complex systems.

Going back to our research project, this foundation was adopted from the very start. It showed itself in the deliverables that were produced, which are presented hereafter under the headings named after some of the aspects that were identified. As usual, the company does not own a proper representation that captures its core knowledge. This leads to confusion and a waste of time and energy, whenever people have to gather and address issues. The effect is worsened when representatives from several organization units, carrying various cognitive universes are called upon. To tackle this situation, Praxeme proposes two kinds of techniques: terminology and semantic modeling.

³ The Enterprise Transformation Manifesto simply defines the “Enterprise System” as “the enterprise that perceives itself as a system” (see on www.enterprisetransformationmanifesto.org).

⁴ Both notions are part of the method, too, but play a secondary role.

The intentional aspect

The intentional aspect of the Enterprise System is made up of objectives (from strategic objectives to operational objectives), terms, requirements, rules, performance indicators. In short, it collects all expressions that cannot be referred to as models, due to the simplicity of their underlying approach. However, these expressions bring value insofar as they convey the fundamental will of the system builders and users. Among these expressions can be found the ultimate goal of the system itself and its ensuing objectives and feature requirements.

One of the first tasks of any project is to clarify the terminology used. To this end, the method recommends collecting available glossaries and developing a thesaurus. The terminological work shows two facets: on the one hand, it passively collects the terms used and analyzes their usages; on the other hand, it elaborates a canonical vocabulary, cleaned from ambiguity and polysemy as much as possible⁵. In so doing, the terminologist paves the way for the modeler. Both disciplines are connected and can help each other when it comes to formulating good definitions. The meaning of a specific term results from its place in the network of terms. Precisely, the best network of terms is the semantic model. Some vague concepts can only receive a proper definition when deduced from the model.

Once the canonical dictionary is available – and ideally confirmed through the semantic model –, we can turn to the requirements and review their formulation. Ideally, every term of a requirement expression should have a corresponding item in the canonical dictionary. This is not enough: the sentences should obey syntactical rules that have been thought through to avoid confusion and to detect all possible options.

These tasks can only be conducted with appropriate tooling. There are many tools available on the market. It is important to consider one major requirement when selecting such tools: the need for dictionary and requirement management tools to smoothly interface with modeling tools. Indeed, we will have to link requirements and terms to modeling elements. The resulting traceability chains provide us with the mechanism to justify the modeling decisions and to check the requirement coverage by the model.

⁵ Incidentally, terminology is a scientific discipline, a branch of linguistics, and it can benefit from a couple of ISO standards.

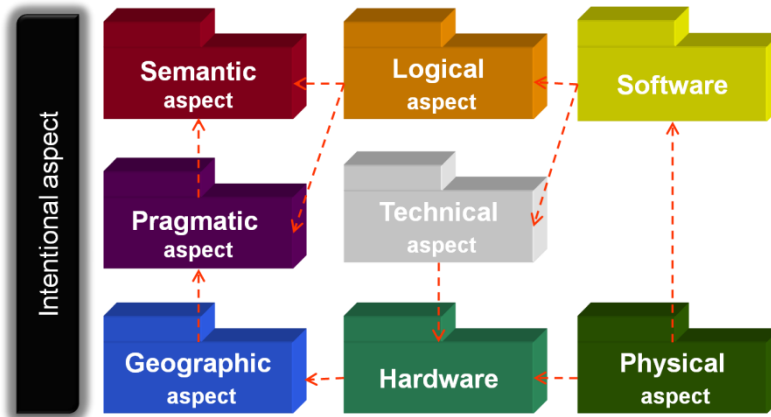


Fig. 1. The Enterprise System Topology (in the form of a UML diagram)

The semantic aspect

Semantics isolate the core business knowledge, that is: knowledge regardless of the way players operate and equipment supports the operations. So, a semantic model is the most abstract of all models, focusing on real objects and concepts and expelling any element that comes from organizational choices or technical solutions. We could think of several means to formally express this knowledge. In our case, we used the semantic modeling technique proposed by Praxeme, which harnesses the object-oriented approach. Alternative techniques include ontologies and formal notations.

We can find some models available on the market, in the field of transportation,⁶ as well as in other sectors⁷. Unfortunately none of these models can be deemed a fully-fledged semantic model. They are essentially data models, sometimes conceptual, most of the time logical, and always far from assuming all the dimensions and reach of business knowledge. Consequently, they can help as input but we still need to undertake the modeling endeavor that captures and clarifies the business knowledge. If we fail to do so, we will not reap the benefits of thorough examination and any investment will not yield its full potential.

⁶ See Transmodel on www.transmodel.org.

⁷ An example is given by the ACORD Framework in the insurance sector. See www.acord.org.

System, where are you?

Everything is not a system

Professionals in the field of transportation commonly use the term “system”, as does everyone else. There is even a norm⁸ that lists the sixteen systems composing a transport system. In these usages, the term “system” does not convey the meaning it has inside engineering and science. It is worth spending some time on this issue because it fuels misunderstanding and makes it more difficult to apply a system approach to the domain of study.

Let us make the assumption that “system” – as our core intellectual tool – is a term and notion that has to be taken care of. A common and widespread usage of the term tends to weaken the notion and to obliterate its real and operative meaning. When the norm speaks of sixteen so-called systems and people get accustomed to seeing these systems in their reality (or, better said, to see reality through these systems), it becomes improbable to exert a systemic approach unless we start raising awareness. At some point, the normative list was coined a “taxonomy” to say what it really was: a list of things we have to deal with. This decision freed the imagination: we were then able to think the system anew, in accordance with the true definition of system. Figure 2 shows how the systems have been put together to form consistent blocks, according to the norm (S1, S2, etc.), in preparation for identifying systems with specified interfaces.

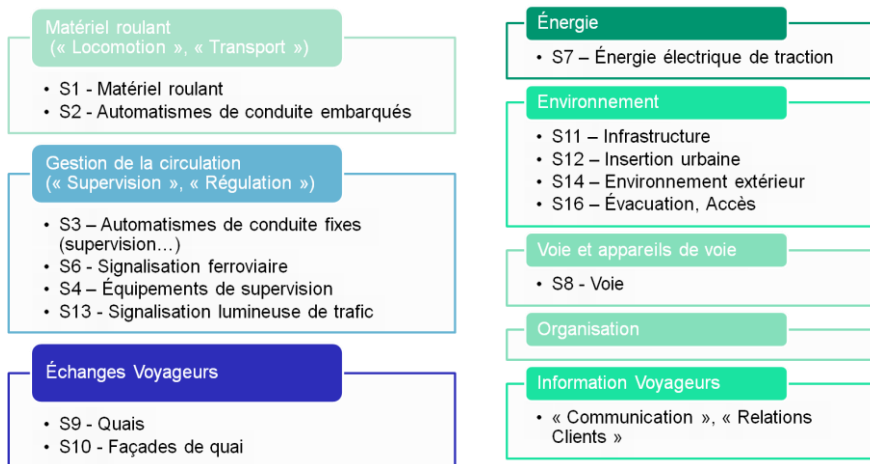


Fig. 2. The systems according to the norm, rearranged as blocks

⁸ Norms issued by the SRTMG (*Service Technique des Remontées Mécaniques et des Transports Guidés*).

The system “S5” covers all communication means. It has been removed from this figure, since the list represents a step toward the logical architecture. Precisely, a logical architecture aims to reveal the interactions among systems and subsystems. Had we introduced the communication system, every system would have been connected to it and this would have masked the dialogues among the systems. This is why the Praxeme method suggests expelling the “middleware” from the logical architecture. Such an architecture endeavor assumes that there will be a solution to connect one point to another at the physical level. The appropriate solution has to be detailed and specified through other kinds of architectures (technical architecture).

A definition of “system” and its implications

A first and brief definition of system can be this one: “Set of interconnected elements, perceived as a whole”. It does not mention any of the features that constitute the analytical and mathematical rigor, as well as the creative power of the systemic approach, but it emphasizes the essence of it, via the term “perceived”. We will not discuss this operational notion here, referring back instead to the vast amounts of published material on this subject. Let us quote Jean-Louis Lemoigne, to whom we owe the definitive progress made notably in his work “Theory of the general system” (1977): “Something identifiable which, in some other thing - its environment -, for something as its project or ultimate goal, does something when running activities, using something internal to its stable form or structure, which transforms itself over time, through evolution.” One can argue this is more of a description than a definition, as it lists corollary features that the notion conveys, instead of expressing its essence, but it provides us with a summary of the implications.

It is necessary, however, to remember the subjective and constructivist value of this notion, summarized very well in this other famous quotation from Henri Poincaré: “The system exists only in the human mind.” This brings us back to the situation of the modeler. For modelers, the system is not an autonomous “being”, given from outside. It is always the result of a commitment, a decision, to cut a more or less coherent part from the substance of reality, upon which we exert our understanding. All modelers must live and breathe this conviction, in order for them to move the line of what is given contingent upon something towards that which becomes a pertinent construct.

The reader may object that this is mere theory. I must admit, yes, but this sort of theory is a preparation for action and arms us with a proper insight into the situation. This lesson proves to be very practical each time a discussion occurs on the choice and scope of a system.

Think the system anew

Let us take a transport system (in our project, it was a given underground line). Our task is to build a representation that is both accurate and efficient: to be a good-enough model, it has to express a maximum number of things with a minimum of terms. Prior to any effort, we have to set aside – or even better discard totally – the usual representations we may have (like the norm discussed above). The best way is to adopt a naïve approach, so as to get rid of preconceived ideas. We identify the mission of the system and its main functions.

The choice of terms is of paramount importance and adopting a renewed vocabulary can be a prerequisite to creativity. For example, if we start saying “drive”, we imply that there must be a driver, thus restricting our design to the manual mode. This is why we favor the more neutral and more open term “transport”, which is closer to the core mission of the system and leaves the options for design open.

Of course, we can always challenge the answer. Why do we have transport? There is always an opportunity for thinking out of the box. Indeed, why do so many people have to commute on a daily basis, generating pollution, wasting time and energy and transforming cities into hell holes? This is a relevant question, but out of the scope of our project and out of the remit of the department involved as well. This is a question for political decision-makers, if they are able to broaden the topic of transportation to consider the much larger system: society itself, with all its dimensions (working hours, organization modes, remote work, culture, etc.).

Words play a huge role here in the following design process and the methodology recommends changing the wording in order to free our imagination. Most of the time, our understanding remains stuck to the current state of things. Using the usual, traditional, technical vocabulary reinforces this alienation. This is why we proposed that the RATP adopts new terms, evoking basic functions, for example: “locomotion system” instead of “rolling material”; “regulator” instead of “command-control station”.

The logical aspect: representation of systems

In the previous chapter, we introduced the intentional and the semantic aspects, respectively the realm of will and of knowledge. The notion of system as a modeling tool emerges from the logical aspect. This aspect plays a very specific role in the method. It is an intermediary aspect between the business reality (business knowledge, organization, business activity) and the artificial solutions we may build (technology, logistics...). As an intermediary aspect, it leaves us free to choose the mindset we apply to it, starting with an appropriate metaphor. The use of metaphor to address this aspect is not meaningless: urbanization for the infor-

mation system, service-oriented architecture for the IT system, functional design, agents, events... We choose our vocabulary because we stand in a situation where we assume the role of designer and creator. Therefore, it is quite natural to resort to the system notion and theory: it is our conscious choice, exactly as a worker selects his/her tool from a toolbox⁹.

The whole system theory is not sufficient to act effectively: we also need to equip ourselves with a notation that will enforce the operative notions. Here, Praxeme proposes a pragmatic option, providing that its main criterion is the ability to connect the models of every aspect according to the Enterprise System Topology schema. This basic solution is UML¹⁰, with the advantages of an international standard and a broad offer for tooling. UML has no notion of system but it is easy to extend it so that we use it to represent systems in a rigorous manner¹¹. When we want to enact a systemic approach via this tooling, we need some key notions that include: visibility, interface, interaction, flow. We also need to establish a clear relation between:

- the systems and subsystem of the logical aspect,
- the classes of the semantic aspect and the roles and use cases of the pragmatic aspect¹²,
- the logical constituents and the physical solutions.

The notation bears stringent rules. The main one: a system can interact with another system only via a specified interface. The dialogues can only convey things that are specified in the interfaces (information, objects, services, signals...).

Figure 3 epitomizes the use of the notation. It shows four systems with some of their interfaces (provided or required). Two systems may be connected if, and only if, they expose the same interface (or compatible interfaces) required for one of them (in this example “Pilotage” for Slq_Regulator) and provided for by the other (Sql_Locomotor). The figure also displays the representation of an interface (in the top left corner) with some of its properties. These properties will be the only elements to be used in the interactions between the systems. This calls for the modeler to explicitly describe every possible interaction and locate them on the model¹³.

⁹ This is not to say that system as a notion may apply only to logical aspect. We can even imagine an approach to an organization that analyzes the organizational unit in terms of systems, meaning real systems in the sense of our definition above.

¹⁰ Unified Modeling Language, a standard issued by the Object Management Group.

¹¹ There is another option: SysML, which is built upon UML and is dedicated to system modeling.

¹² While the semantic aspect deals with business and real objects, the pragmatic aspect deals with activity (from processes to use cases) and actors (roles, organization...).

¹³ Note about the notation: UML does not allow us to connect components at a type level, as shown in this picture. Some tools offer this possibility.

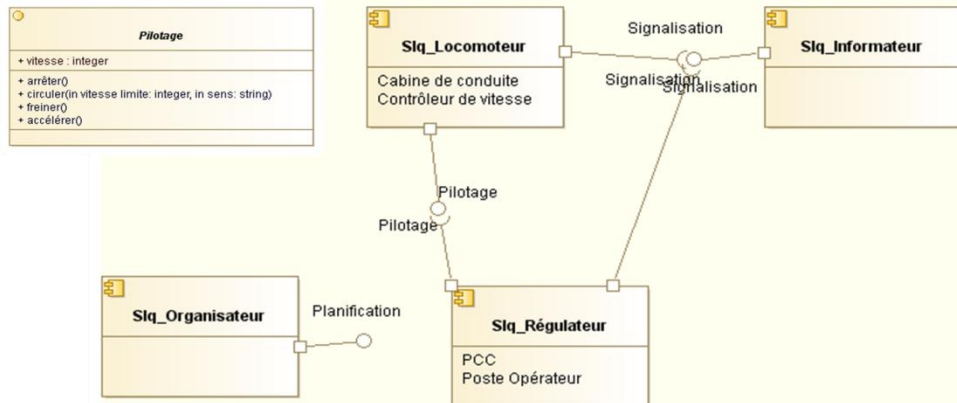


Fig. 3. Representation of systems and their connections (example)

The human being and the system thing

One of the findings when designing the transport system from a logical architecture perspective has been that, sometimes, we ought to put the players inside the system, even when it is an artificial system. Doing otherwise would impair our ability to design systems in accordance with the encapsulation principle. This is particularly obvious in the case of a vehicle. The company handles two transportation modes: traditional, with a driver onboard and automatic (e.g., line 14 of the Paris metro). In some cases, both modes are mixed (e.g., line 1). From the traffic regulator's perspective, the mobile system should be described the same way, whichever the mode. When we are not able to do so, we increase the complexity of the representation and we fail in our bid to construct a generic and convergent architecture. Therefore, the mobile system exposes an interface with all required services to steer the vehicle. These services are to be interpreted from inside the system, depending on the solution implemented (manual or automated). In so doing, we preserve the genericity of the architecture; the regulator system can be designed in such a way that it can cover every case. Had the command-control station been designed following this rule, the transition of the underground service from manual to automated mode would have been facilitated.

Some people may feel uncomfortable with this idea that the human actor is "lowered" and reduced to the level of a component. At least two categories of people express this concern:

- People with an IT background (you have never seen human being inside a computer!);

- Naïve humanists who rebuke any attempt to formalize the interaction between man and machine (but we will reply that there is another aspect of the architecture framework that is dedicated to the analysis and design of human interaction).

Here, we are dealing with the logical aspect, and we need to put all kinds of resources together to make the system run. If we study an artificial system as being separate from the human system, we miss the true meaning and complexity of the system. This is why the method recommends modeling human beings among the material artifacts, thereby anticipating that the former could revert to the latter from time to time. It is the only way to comply with the encapsulation principle and, thus, to simplify the design.

The system boundaries: between arbitrary and arbitration

A critical question when designing systems is one about the breakdown structure. A spontaneous and unchallenged representation is not always the best one. Let us take the example of the track. The semantic model attaches many properties to the Track class, notably its states: occupied, free... These features certainly belong to the semantics of a track. It is a no brainer, as far as semantic modeling is concerned. But when it comes to logical modeling and to delineating the subsystems, how should we proceed? On the one hand, the tracks are part of the infrastructure, which is a separate subsystem (with a notion of dedicated accountability). On the other hand, the state of a track is checked and displayed thanks to signals, which pertain to the means of signaling. There is a great temptation to create a “signaling” subsystem, inasmuch as it reflects the human organization (there is a department devoted to this topic). Choosing this spontaneous option leads to breaking the semantics of a track, scattering it among at least two subsystems. A better option would be to adopt the view that signals intervene only as means to express the inner states of objects and, as such, they should be modeled close to these objects. Unfortunately, this purist idea goes against the usual way of perceiving things and organizing people; it hits an immutable factor that habits have engendered. It is not to say that the culture of the company is impervious to change, but we need time for this kind of cultural shift to happen. Modeling attempts always unravel such knots. If not, this is a sign that modeling has not dug deeply enough. To conclude with this example, identifying systems and establishing their boundaries are acts subjected to uncertainty and a topic for debate. There may be a best solution but it will not necessarily be accepted or, to put it in other terms, one may have the feeling that there still remains a sort of arbitrary approach when delineating the systems.

A second example will show that the affair can be even trickier. Let us examine the classical dichotomy Product *versus* Production (or system-to-be-designed *versus*

sus the socio-technical system, i.e., the project that produces the product). Where should maintenance stand? It would make sense to put it on either side:

- Maintenance contributes to transforming the product or to creating a new version of the product, or a new product. As such, we can justifiably decide that maintenance pertains to production. By the way, the total cost of ownership includes the maintenance costs, and the technical debt should be estimated once the project starts.
- However, there is another way to see things; this will appear if we put ourselves in the customer's shoes. From the client's perspective, the product system encompasses every element, every service and interaction that makes the product work. And the product works provided that the fees are paid, that the call-center answers the request, that the after-sales service repairs the failed parts, and so on and so forth. Following this analysis, maintenance is part of the product.

How can we decide between these two options? They are equally logical and, even though we fully develop both architecture scenarios, it is not sure that we will be able to differentiate between them with quantitative metrics. Nevertheless, their implications differ dramatically. I bet that the second option would drive a significant improvement in terms of quality of service and corporate image. But we face what we could coin arbitrariness or, at least, liberty: it's up to us, as architects and designers, to choose. When confronted with several equivalent options, the decision falls to us. . With liberty comes duty.

Both examples (the signaling system and the maintenance case) share the sense of arbitrariness when drawing the boundaries of the systems. The second example is more about the organization and the discussion will be positioned in the pragmatic aspect, in Praxeme terms. On the contrary, the first example – taken from our case study – epitomizes the decisions architects have to make through the logical architecture. At this level, it is worth noticing an amazing phenomenon that tempers the debate: the subsystem boundaries are not so important! At least, they are less important than the interfaces that we bring out. At the end of the day, what really matters are not so much the boundaries of the subsystems, than the set of interfaces that compose the system and that ensure the interactions and operations. This is the outcome we expect from the logical design. We can vary the way we decompose the system, but we will always need the same set of interfaces. We can dispatch the interfaces through several scenarios of architecture. A given interface may simply appear at various levels depending on the architecture scenario. This phenomenon holds true within certain limits, of course. It may be more easily understood when we examine the impact of the logical model. First and foremost, the logical model can be seen as the catalog of contracts, expressed through the interfaces (in the sense given by UML). Specific solutions, as they are delivered by the industry, refer to these contracts. In UML terms, we would say that the solution components implement the interfaces. This task occurs in the material aspect of the Enterprise System (tantamount to logistics). At this level, the material struc-

ture is not necessarily isomorphic to the logical structure. What is mandatory is that the material or physical solution concretely translates every required interface.

How can models anticipate issues in the real world?

The second phase of the research project aimed to demonstrate the capacity of the method to provide insights into the designed or developed systems and to anticipate potential issues.

The case study

The case study was proposed by the steering committee. In the past, the signaling system was based on incandescent bulbs. As this technology was known to be subject to failure, the company developed a control device that informed the traffic regulator should an incident arise. As a result, the regulator in the central command post always had the same perception as the driver on the ground. Then came LED technology. In a general excess of optimism, the industry suppliers and the company decided that, due to the resilience of this technology, the control device was not necessary anymore. The migration started. Unfortunately the damaged filament was not the only cause of failure. For a signal – even an LED display – to work, it needs electricity! And sometimes there are problems with the electricity network. Because the new configuration gets rid of the feedback loop, it results in a situation where the traffic regulator has a different view of the reality perceived by the drivers. When drivers find themselves in front of a switched-off signal, they have to stop, even though they may know this is only a minor issue and the track is free. The regulation imposes such mandatory behavior. At the same time, the traffic regulator still thinks that the situation is fine and that the train is able to move on. This results in a delay before the situation is handled correctly. The full picture is a little more complicated, since it also contains a device that automatically stops the train in case the driver does not pay attention to the signal (see figure 4).

The question posed was: would a proper modeling approach have emphasized this issue?

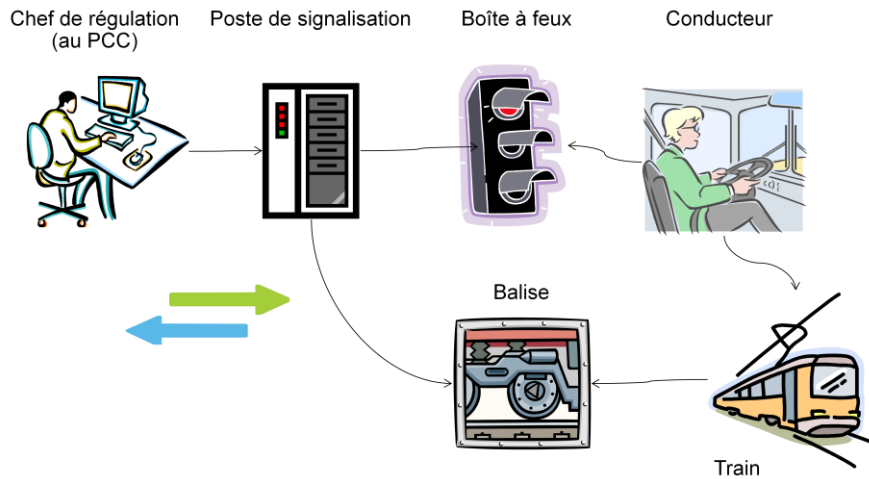


Fig. 4. The element of the problem (intuitive representation)

The problem solving approach and the techniques used

The first task is to represent the components involved in the case. To this end, we used the UML component symbol. The semantics of component in UML are restricted to software components, but they have been extended to cover all kinds of things (in SysML for example).

Figure 5 fits the elements of the problem together. It also displays the interfaces between these elements. The picture is certainly a simple one, but it took a while to stabilize it and it required us to uncover all implicit knowledge. For instance, we discovered that the lights were plugged into the signaling post and not directly into the electricity network. We had to make the interactions clear too.

The interfaces are typified, and the customized notation adopts the color code of the Enterprise System Topology: every aspect is given a special color. Many interfaces come from the logical aspect and, thus, express command-control interactions. Their symbols are colored in orange, which is the color of the logical aspect. The physical interfaces – here with the electricity network – are blue. Last of all, we have to take into account interactions between the system and the human actors. To understand the overall behavior of the transport system, we have to introduce the behavior of the driver in front of the signal. Human perception and the attached states need to be included in the model. The human-machine interface is represented with the color of the pragmatic aspect, the aspect of the Enterprise System dedicated to human beings and to organization.

That way, the model accounts for all the dimensions of the problem. These dimensions are weaved together at the core of the model, still being disentangled according to the “separation of concerns” principle.

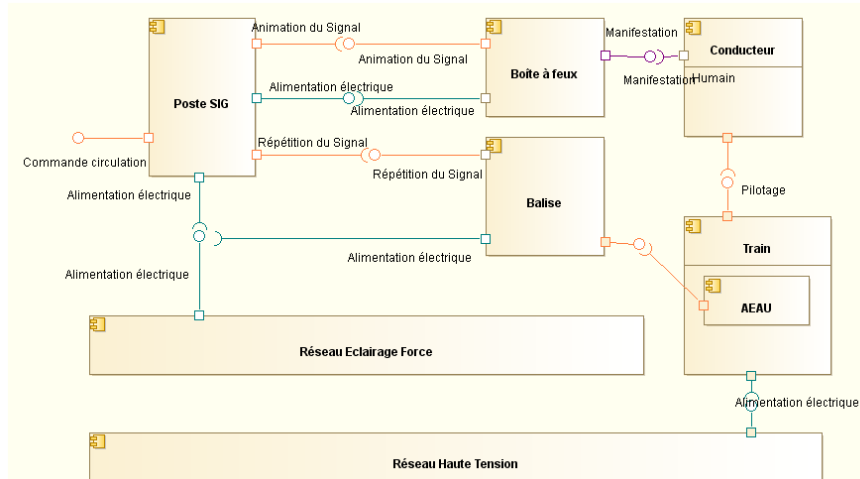


Fig. 5. Representation of the components that come into play in the case study (UML)

Knowing the elements, we have to examine their behavior. The state diagram below (figure 6) illustrates this modeling technique, which equips us with a powerful tool to analyze complex systems and their dynamics. This technique reveals itself to be not too difficult in practice because it applies to objects, at the scale of a single object. It consists in capturing the adjectives and phrases used in connection to the object name, then in linking them in a logical manner to reflect the behavior of the object.

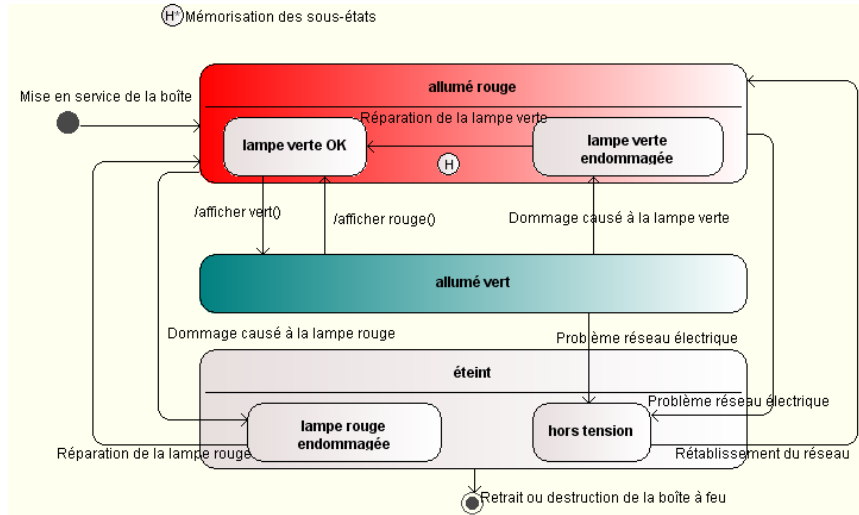


Fig. 6. The state diagram of a signal box

Now we are ready for the final act: with sequence diagrams, we document several scenarios that involve the component. The example below details the case of the LED device without feedback, when the regulator sends the authorization to move ahead but the signal box is out of order. In this configuration, the failure provokes two discrepancies, leading to a traffic incident.

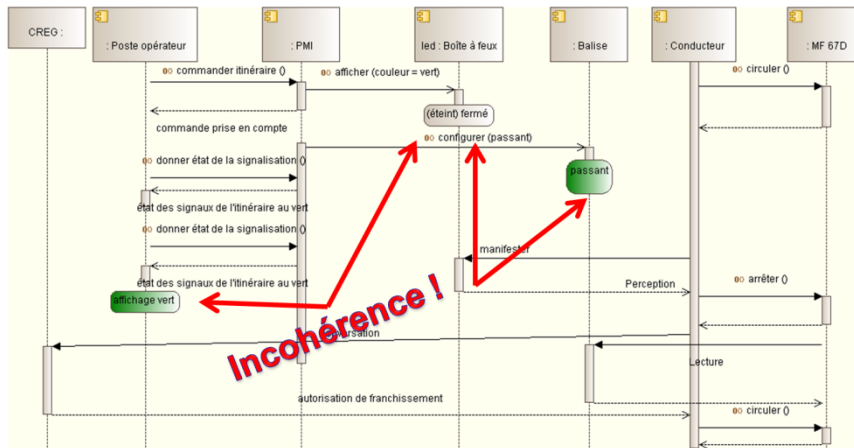


Fig. 7. A sequence diagram that helps to identify the issues in a special script

The outcome

We used several sequence diagrams to analyze execution scenarios in various configurations: the previous solution with incandescent bulbs and feedback device; the new solution with LED; the LED solution with a new feedback device to be deployed... Not only did this proof of concept confirm the value of modeling in these matters, but it also revealed that the current solution being deployed does not work perfectly. Reading the corresponding sequence diagram, it became obvious that this was a partial solution – despite its cost – and that it will not prevent some incidents from happening.

This experience shed light on the current status of the documentation and on the potential of a more rigorous approach based on modeling techniques. The participants in the project were familiar neither with engineering nor with modeling, but were aware of the weaknesses and risks related to the status quo. At the end, they were convinced Praxeme offers an appropriate approach to physical systems, able to cast new light on design, to stimulate innovation and to tackle specific issues, filling the spectrum from overall architecture to detailed design.

From this specific example we drew a more general conclusion. The following principles were adopted that condense the essence of requirements, from the system owner’s perspective:

- Whatever the system, subsystem or part at any level, the solution should guarantee constant feedback throughout the system. This high-level requirement is represented by a “synchronic” message in UML, as shown on figure 8 (with adapted semantics, not to be confused with its meaning in IT).
- Responsibilities are stated in terms of aspects and of systemic levels.

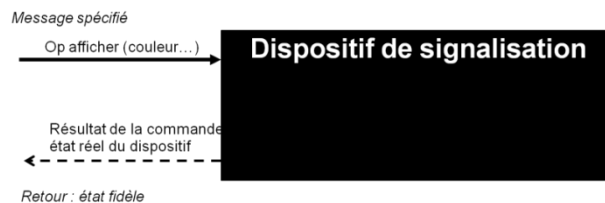


Fig. 8. A way to represent the generic feedback requirement and to assess compliance

In terms of responsibilities

A major concern in the company is with regard to the clarification of responsibilities between the system owners, the other departments and the component providers. With this aim in mind, the method frame can be used as a grid to revisit the process and to specify the deliverables with quality criteria and clear responsibilities.

Using the architecture framework (the Enterprise System Topology) helps to assign responsibilities in terms of models, rather than assigning them all along a process. The easy way consists in fixing responsibilities per aspect. Obviously, the models of intentional, semantic, pragmatic and geographic aspects are under the system owner's remit. As regards the logical aspect, things get more subtle. We could say that the system owner is the only person responsible for the logical model, while the industry providers take care of the material aspect. This could work in some contexts but not here. The logical model is far too detailed and the owner refuses to dive into excruciating details. As a result, we can specify responsibilities by fixing a level of detail for the modeling. The system owner's responsibilities cover the big picture, the overall design, and it goes down to the level where industrial partners are identified. At this level, components are black boxes in the eyes of the system owner. These boxes are perfectly specified to such a point that test cases are attached to them. Then the industrial provider has to satisfy the specification and to comply with the modeling approach. To the provider, the component is a white box, designed so that it fits the upper-level design, as shown in figure 9.

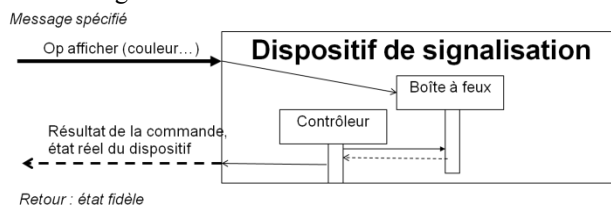


Fig. 9. White box design with arrangement of connection types to meet requirements (on a simplified sequence diagram)

Conclusion

Praxeme can apply to physical systems¹⁴, provided that some adjustments are made on its framework. We are carrying out an ongoing reflection and collective discussion inside the Praxeme Institute in order to review the Enterprise System Topology. Our goal is to generalize the architecture frame, so as to apply it to any kind of system without further adjustment. We also seek to build it on a solid theoretical basis, and we call for the support of the scientific community to this end. This endeavor will lead to Praxeme version 2.

¹⁴ In 2003, the first application of Praxeme was to unmanned aircraft vehicle systems, and SAGEM was the first of the contributors to join the initiative for an open method. There have been experiences in the field of weapon systems too (Thales, DGA, Dassault).

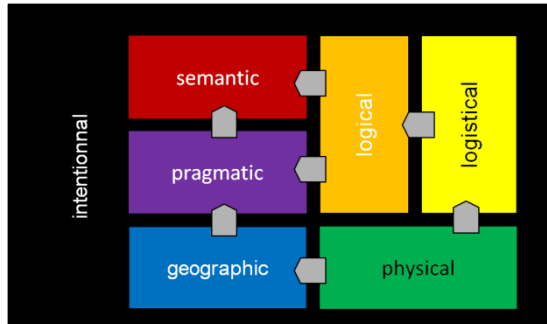


Fig. 10. The future version of the Enterprise System Topology (informal representation)

Modeling is a powerful tool that helps us to anticipate and avoid issues in the real world. This is common wisdom, but only inside the community of modelers, architects and system practitioners. Nevertheless, it has to be constantly reasserted against the common mood that affects the management sphere and ruins any attempt to take the necessary time for thorough reflection.

This demonstration has been done in the context of the department in charge of the transport systems at RATP. The question now is whether this organization is willing to follow through with the demonstration and to cope with the growing complexity of its systems. Will it shy away in the face of technical difficulties, organizational deadlock and cultural resistance? Or will it be able to build on these conclusions and adopt a new approach, in spite of its fear that it lacks the competencies and resources to undergo this cultural shift? Ten million commuters are expecting them to!

Bibliography

- Aiguier M., Le Gall P., Mabrouki M.: A formal denotation of complex systems: how to use algebraic refinement to deal with complexity of systems, Technical Reports available on www.epigenomique.genopole.fr
- Aiguier M., Le Gall P., Mabrouki M.: Complex software systems : Formalization and Applications, International Journal on Advances in Software , 2(1):47-62 (2009)
- Apostel Léo: Syntaxe, Sémantique et Pragmatique, in Logique et Connaissance Scientifique dir. Jean Piaget, coll. La Pléiade, Gallimard (1967)
- Bonnet P., Detavernier J.-M., Vauquier D.: Sustainable IT Architecture, Wiley (2009)
- Caseau Y., Krob D. et Peyronnet S.: Complexité des systèmes d'information : une famille de mesures de la complexité scalaire d'un schéma d'architecture, Génie logiciel (2007)
- Chingareva-Slavine E., Sémiotique, linguistique et modélisation, Lavoisier (2003)
- Depecker L.: Entre signe et concept, Eléments de terminologie générale, Presses Sorbonne Nouvelle (2003)

- Enterprise Transformation Manifesto, www.enterprisetransformationmanifesto.org (“In the face of complexity, this manifesto articulates core principles and offers an escape from confusion, gloom and doom. It aims to reinforce our ability to act.”)
- Garajedaghi J.: Systems Thinking, Butterworth-Heinemann (2006)
- ISO: Norme internationale ISO 704, Travail terminologique-Principes et méthodes (français et anglais), Afnor, 3e édition, décembre 2009, 65 p. (1ère édition 1990, 2e édition 2000)
- Le Moigne J.L.: La Théorie du système general, PUF (1984)
- Meinadier J.P.: Ingénierie et intégration des systèmes, Hermès (1998)
- Object Management Group, OMG Systems Modeling Language (SysML) OMG (2010)
- Object Management Group, OMG Unified Modeling Language (UML), OMG (2010)
- Poincaré H.: Science et method, éd. Flammarion (1908)
- Vauquier D.: Praxeme methodological guides, www.praxeme.org
- Vauquier D.: Développement Orienté Objet, Eyrolles, 1993
- Walliser B.: Systèmes et modèles, Editions du Seuil (1977)