An Executable Model Driven Framework for Enterprise Architecture Application to the Smart Grids Context

Rachida Seghiri
EDF R&D
LRI, Université Paris-Saclay
91405 Orsay, France
rachida.seghiri@edf.fr

Frédéric Boulanger
CentraleSupélec
LRI , Université Paris-Saclay
91405 Orsay, France
frederic.boulanger@lri.fr

Claire Lecocq
Institut Mines-Telecom
Telecom SudParis
CNRS UMR 5157, SAMOVAR
91011 Evry, France
claire.lecocq@telecom-sudparis.eu

Vincent Godefroy
EDF R&D
91141 Clamart, France
vincent.godefroy@edf.fr

Abstract

We propose a framework that allows for modeling Enterprise Architectures (EA) in order to automate analysis, prediction, simulation, and thus to address the key issue of business/IT alignment. Smart Grids are power grids enabled with Information and Communication Technologies. Extensive studies try to foresee the impacts of Smart Grids on electric components, telecommunication infrastructure, and industrial automation and IT. Our message is that Smart Grids have also impacts on the overall EA of grids operators. Therefore, our framework enables stakeholders to validate and criticize their modeling choices for the EA in the context of Smart Grids. It is a multi-view framework regarding three aspects — information, processes, and goals —for each view. We add an integration view to ensure inter- and intra-view consistency. We rely on Model Driven Engineering (MDE) techniques to ease the holistic modeling and analysis of enterprise systems. Finally, we show the utility of our approach by applying it on a Smart Grid case study.

1. Introduction

A Smart Grid is a power grid enabled with information and communication technologies in order to optimize the production, distribution and consumption of electric power [25]. Smart Grids represent a major paradigm shift for organizations dealing with electric power, especially operators, and has a significant impact on their enterprise systems.

On one hand, Smart Grids lead the involved organizations to consider their own business models and the underlying IT, and thus the whole EA, for a better control of power grids. In fact, EA consists in "the explicit description and documentation of the relationships among business and management processes and information technology" [12].

On the other hand, stakeholders may have different interests (business strategies, environmental issues, standards, technologies, etc.). Enterprise architectures allow them for an effective alignment of these interests.

Simulation is a well-known method used to validate and criticize systems design in the early stages of development. Hence, we aim to enable experts involved in designing an EA that addresses Smart Grids to have direct insights into models through simulation. We identify three main obstacles to overcome in order to achieve this goal.

First, modeling comes prior to any simulation activity. There are many existing EA frameworks. These frameworks are based on multi-view modeling approaches. Such approaches are essential for the comprehension of complex systems like the EA of organizations dealing with Smart Grids. However, the models produced are usually intended to document enterprise solutions and to communicate about them. Modern organizations have to align their strategy and business goals with their IT infrastructure to maximize their benefits. Expressing explicit links between EA views and automating their exploitation is a way to achieve effective alignment.

Second, Smart Grids involve several experts as they combine electrical engineering and IT engineering. Therefore, a model must meet two criteria: (1) it is understandable to all the stakeholders involved; (2) it is based on sustainable standards. Smart Grids experts are effectively adopting standards like UML [24]. Nonetheless, EA must effectively integrate these standards.

Finally, a model's ability to be executed is a sine qua non condition to simulate it. As mentioned, models produced to express artifacts of an EA lack formal semantics since these models are aimed at documentation and communication. So far, the executability of a language is closely linked to the way its semantics is formalized. Hence, expressing the formal semantics of a modeling language remains a key challenge for EA.

To address the obstacles above, we propose an integrative approach using several views to model and simulate an EA. In our approach, we consider models as first class citizens as they are "productive" instead of "contemplative" models. This is precisely the leitmotiv of Model Driven Engineering (MDE) [4]. We take advantage of MDE principles and techniques such as model transformations and executable models in order to align an organization's business goals with its IT features before going through simulation.

The paper is structured as follows. Section 2 depicts the theoretical background regarding MDE concepts and techniques, EA frameworks and the simulation of an EA. Section 3 describes the industrial issues inherent to Smart Grids. In section 4, we detail our approach and its application to a Smart Grid case study. Section 5 deals with the related work. Section 6 concludes the paper and gives some perspectives to our work.

2. Theoretical background

Smart Grids and related EA are, in essence, complex systems [17]. MDE has proven its ability to tackle such systems through extensive work [9]. The first part of this section deals with MDE. Our purpose is to model and simulate the EA of Smart Grids. We present the main views used for EA modeling in the second part. Finally, we specify the objectives of EA simulation and the available means to achieve it.

2.1. Model Driven Engineering

MDE promotes the "*everything is a model*" paradigm [4]. Indeed, it is a model driven approach that covers the whole life cycle of a software system using "productive" models [3], i.e., machine-processable models (simulation, validation verification, code generation, etc.). Models are first-class citizens in MDE. Although the definition of what a model has raised some debate, a common definition is: *"A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system"* [5]. This definition induces the first basic relation of MDE. Indeed a given system is *RepresentedBy* by a model, which only selects the relevant details from that system in order to answer the modeler's questions.

Furthermore, MDE makes it possible to capture the business logic through metamodels. Indeed a metamodel describes all the handled concepts in addition to their relationships and constraints. All models used should conform to this business logic and so to the metamodel. This brings us to the second basic relation of MDE, namely *ConformsTo*, which connects a model to its metamodel [8].

The main interest of MDE is to allow for an automated manipulation of models through model transformations. A common definition of a model transformation is given by [15]: *"A model transformation is the automatic generation of a target model from a source model, according to a transformation description"*.

As illustrated in Figure 1**Erreur ! Nous n'avons pas trouvé la source du renvoi.**, the transformation description is given at the metamodel level. A transformation engine performs this transformation. Model transformations lie at the heart of a Model Driven approach. MDE promotes their use throughout the whole life cycle of a software system such as for refinement, composition, analysis, or simulation. Likewise, model transformations make it possible to quickly align IT with business needs and easily capitalize business knowledge in the metamodels.

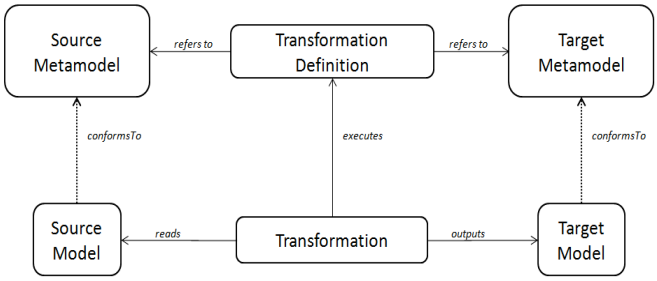

Figure 1. Model transformation components

Nevertheless, the extensive range of the handled models should be consistent and representative of the final system. Therefore, maintaining consistency across models and their transformations is an active area of research in MDE. One way to achieve this purpose is to type entry models of transformations through Model Typing [22]. Model Typing offers to increase the reuse of transformations by highlighting the common characteristics of models. Figure 2**Erreur ! Nous n'avons pas trouvé la source du renvoi.** illustrates this principle. The model m_A is typed by MT_A, and a model transformation $t_{A \to C}$ transforms m_A into m_C. If you need to transform another model m_B into m_C, you may want to reuse the same model transformation. If MT_B is declared as a subtype of MT_A, you will be able to perform $t_{A \to C}$ to do so. In this way, Model Typing may be very useful for EA. We give some examples of its potential use in section 4. Moreover, it is a tooled approach. Kermeta [11] is a metamodeling language that implements Model Typing.

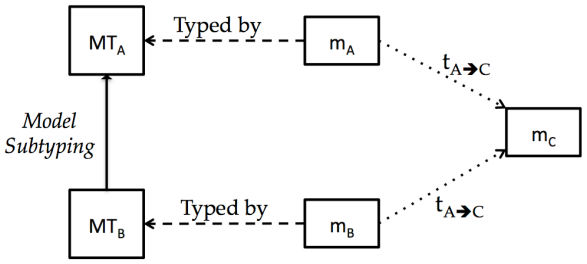

Figure 2. Model Typing principle

2.2. Enterprise Architecture

Enterprise Architecture is an effective way to catch an enterprise system in its current and desired states. This description should reflect the concerns of business analysts, data architects, functional architects, application architects, technical experts, etc. Adopting a monolithic vision of EA, especially in the Smart Grid context, is unsuitable to build an adaptive description given the complexity of these systems and the large number of involved stakeholders.

For this reason, several frameworks are based on a multi-view approach in which a view conforms to the perspective of a particular stakeholder. Such an approach helps to handle complexity by separating the concerns of stakeholders in different perspectives.

Zachman's framework and The Open Group Architectural Framework (TOGAF) are among the most popular frameworks adopting a multi-view approach [1]. The Smart Grid Architecture Model (SGAM) framework [24] is also worth mentioning. It addresses the architecture of Smart Grids by combining three domains: information systems, power grids, and telecommunication networks.

Views slightly differ between frameworks. But, generally speaking, there are five main views:

- **Business view:** This view reflects the business vision including business objects, processes and actors;
- **Information view:** This view describes information required to perform business processes;
- **Functional view:** This view describes functional blocks that realize business processes and functional processes that refine business processes. This modular structure ensures flexibility and adaptability while meeting the organization's needs;
- **Application view:** This view is divided into applicative modules. Each one implements one or more functional blocks;
- **Technical view:** This view describes the technical infrastructure on which applications are deployed (hardware and telecommunication networks).

Moreover, these frameworks hierarchically organize the different views according to the *"IT follows business"* principle: starting with the business perspective, deriving it progressively into the technical infrastructure via information, functions, and applications. Nonetheless, the information view deserves special attention. Indeed, information is modeled as an aspect of all other views (Zachman) or separately in a dedicated view (TOGAF, SGAM).

The models describing these views are usually "contemplative" models. Mainly used for communication and documentation purpose, these models are disembodied from their implementation. Moreover, languages used to model EA features, such as ArchiMate and UML, offer extended concepts to express the modelers' needs, but lack formal and rigorous semantics that allows for automation.

Automated manipulation facilitates business and IT alignment. Indeed, the use of executable models improves consistency among views and eases their understanding by the stakeholders who can simulate these models in the early design stages.

2.3. EA simulation

According to Shannon [21], simulation is *"the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system."*

Regardless of the application field, simulation is a powerful tool to evaluate the modelers' choices for the system structure and behavior. Simulation may take the form of a model animation and the study of the behavior of this model according to the inputs.

Simulating EA in the context of Smart Grids is crucial as they are in constant and swift change: fluid regulatory frameworks, emergence of new partners, heterogeneous interactions with customers through smart meters, smart phones, digital tablets, etc.

Model executability is necessary to reach model simulation. Execution capabilities of models make the models more comprehensible for experts and avoid ambiguities caused by purely contemplative models.

Model execution is made possible by defining execution semantics for the language that expresses models. The semantics of a language expresses the meaning of the concepts and their arrangements when instantiated at the model level [18]. The construction of the semantics depends on the pursued goal: simulation, code execution, verification, etc. Expressing a language semantics is the focus of intensive research especially in the field of formal languages [13].

The IBM manifesto [6] attests that the three main axes of MDE are: (1) open standards, (2) automation, and (3) direct representation. Given these axes, we adopt standardized and executable languages that are understandable to the stakeholders involved in EA in a

Smart Grids context. We identify several languages that satisfy these criteria:

- A subset of UML diagrams limited to class diagram and activity diagram has henceforth an execution semantics formalized by the fUML (foundational UML) standard. Class diagrams are suitable for the representation of information models while activity diagrams are suitable for the representation of the behaviors expected;
- BPMN is a standardized graphical modeling language. It allows for the representation of most aspects of a business process within a unique diagram. BPMN has a well-defined execution semantics. Thus, a variety of tools for the simulation of business processes implement this language;
- The Object Constraint Language (OCL) is a standardized textual language for expressing constraints on UML diagrams in order to model properties that are difficult to capture in UML models. The execution of OCL is made possible by model transformation (targeting a lower level language like MiniZinc [19]) or by using it at the metamodel level (OCLinEcore [27]).

3. Industrial background

In this section, we describe the industrial issues we address before introducing the Smart Grid case study on which we test our approach.

3.1. Industrial issues

A good operation of power grids relies on the balance between consumption and production. Henceforth, Smart Grids are essential to maintain this balance and to handle the massive penetration of electric vehicles and renewable energy sources. Smart Grids provide automatic and real-time energy management via sensors and remotely controlled checkpoints.

In essence, Smart Grids bring profound changes to the IT that drives them and thus to the whole EA: new flows of information sent from the power grid, new stakeholders such as decentralized energy resources (wind farms, photovoltaic panels), new communication devices like smart meters, the need of conformance to the new European regulations and directives [26], new usages (electric vehicle, connected home).

To handle the emerging paradigms, experts are developing new use cases that need to be tested and validated before their final adoption. Various demonstrators are deployed in the field. These pilot projects allow for the conduct of experiments in real conditions to test different functions and services.

For instance, InfiniDrive and Ventea are two French demonstrators. InfiniDrive controls an electric vehicle charging infrastructure. Ventea handles a rural grid with high wind capacity penetration. However, physical demonstrators force the grid operator to enroll industrial and/or domestic customers who are willing to install test equipment at home. Moreover, their operation is limited by current regulations. Also, their implementation is often expensive and time consuming.

Besides these demonstrators, full-scale test-grids allow for the evaluation of new equipment before its deployment on the real grid. ERDF (Électricité Réseau Distribution France), a major French distribution system operator, maintains Concept Grids for this purpose. Indeed, it is possible to conduct stress tests in disturbed conditions that would be impossible to perform on a real grid, with real customers. Nevertheless, the small size of these networks remains limiting.

Simulation makes it possible to overcome these limitations. Such a simulation includes the three domains of a Smart Grid: electrical infrastructure (transformers, lines, loads, sources), telecommunication infrastructure (mobile network, BPL) and, of course, IT. Specialized simulators for power grids (EMTP-RV, Dymola, PowerFactory, Eurostag, etc.) and for telecommunications (OPNET, NS-3 OMNeT ++, etc.) have already proven how powerful simulation is in their respective fields. Nonetheless, EA is often reduced to its IT component and relegated to mere set-points calculation written in Matlab or C++ [20].

Our ambition is to show how the simulation of EA models can also be effectively used by experts involved in Smart Grids development. Indeed, Smart Grids have an impact not only on the electric and telecommunication infrastructure of a given system operator but also on its interactions with customers, its relations with alternative power providers, its strategic partners, its internal processes, its branding, its services, its relations with local authorities, etc.

3.2. Case study: Management of an electric vehicles fleet

New applications of electric technologies are emerging including electric mobility. Public authorities estimate that approximately two million electric vehicles will be on the French roads by 2020. This ambition is motivated by the objective of the French government to reduce greenhouse gas emissions by four over four decades.

Several R&D projects aim to handle the impact of the increasing number of electric vehicles on distribution power grids. Indeed, full charge of an electric vehicle with 150 km autonomy is equivalent in terms of power demand to:

- A water heater if the battery charges in 8 hours (normal charging);

- A building if the battery charges in 1 hour (accelerated charging);
- An urban area if the battery charges in 3 min (fast charging).

Maintaining balance between supply and demand is a key issue to ensure a high quality delivery of electric power. For this purpose, when demand increases, power grid operators start coal and oil plants, which are costly and emit CO_2. Ongoing experiments are testing new methods to avoid supply/demand constraints on power grids without starting costly and polluting plants. For instance, La Poste (French mail office) and ERDF implemented InfiniDrive to optimize the charging system of the electric vehicles fleet of La Poste. Optimization algorithms and related charging infrastructures aim to minimize the concentration of accelerated and fast charging during peak hours while taking into account incentive tariff during off-peak hours.

Engineers use simulation to improve the design of these charging systems. However, electric mobility involves a paradigm shift not only for the grid infrastructure but also for higher company processes and constraints. Indeed, the electric vehicle is limited by its autonomy so it cannot be used for any tour. Also, charging an electric vehicle is different from filling the tank of a combustion vehicle (time of charging, charging station availability).

As a result, the massive use of electric vehicles deeply affects the whole enterprise system: it has an impact on stakeholders like fleet managers, on agents who drive the electric vehicles, on business processes (tour optimization, allocation of vehicles to these tours, agent's tours while using an electric vehicle, etc.), on information (new business objects come out like electric vehicles, charging points, etc.), and on applications (new applications have to be deployed and integrated with existing ones).

French electric grid operators used to have a relatively slow evolution of their business and technical environment. This paradigm is changing as Smart Grids bring broad, deep and rapid shifts on organizational vision. To anticipate the resulting EA transformations, we propose to model, analyze and simulate EA by automating validation, deployment of an enterprise solution while maintaining consistency across EA views.

Hereafter, we present our executable Model Driven approach for EA while maintaining consistency across views. This approach allows us to perform automated analysis and simulation through the use of executable models, Model Typing and model transformations.

4. Approach and its application to the case study

In this section, we present our approach and we apply it on the case study of managing an electric vehicles fleet. Our contribution is twofold. First, we propose a multi-view framework for EA with an additional view: the integration view. This view aims to address consistency and alignment issues. Second, we use executable and standardized languages from MDE to model and simulate EA.

4.1. General approach

Our approach is based on multi-view modeling. We identify four views addressing concerns of experts and partners involved in modeling and validating EA features.

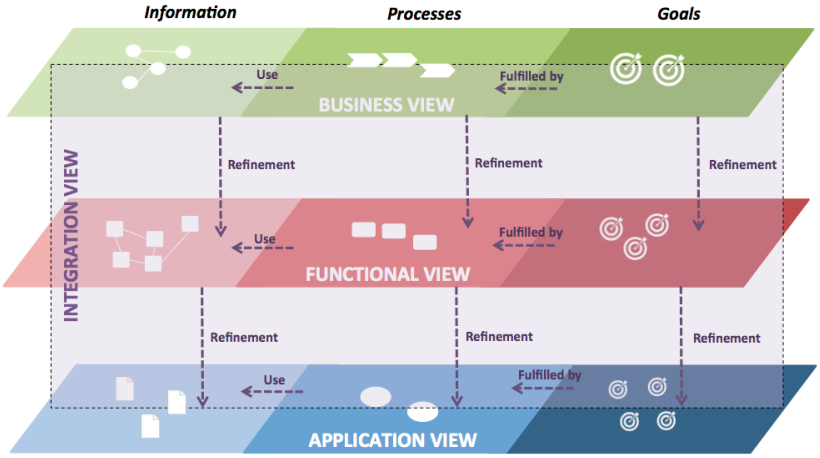

Figure 3. Proposed approach

In our approach, we consider only business, functional, and application views (Figure 3). But we

aim to generalize it to the technical view as well. Most frameworks commonly use these views.

- We do not model information in a dedicated view. We explicit information as an aspect of the other views as recommended by Zachman's framework. In fact, information is spread in the business, functional, and application views:

- **Information aspect of the business view**

This aspect represents the major business concepts handled by the business processes. This model is not very prone to change, unless a significant change occurs in business practices. It is involved in the division into blocks of the processes aspect of the functional view;

- **Information aspect of the functional view**

This aspect represents the functional data type and gives a detailed description of data handled by functional blocks. It refines the business concepts by giving their types. It models the data properties, relationships, etc.;

- **Information aspect of the application view**

This aspect represents the application data model, which highly depends on the chosen software. It refines the functional data and gives the required data formats that are compliant with the application modules.

In addition, we model the dynamics of views in the processes aspect. Thus, for each view, we differentiate the flowing processes aspects:

- **Processes aspect of the business view**

Typical artifacts of this aspect are business processes and involved actors. The use of executable and standard formalisms is highly recommended to ease automated analysis and simulation. UML activities diagrams and BPMN are good candidates. However, Domain Specific Modeling Languages can also be used for this aspect;

- **Processes aspect of the functional view**

This aspect reflects the perspective of the functional architect and describes the functions that realize business processes and their orchestration as functional processes. These functions are gathered into blocks. A unique functional block handles each business concept identified in the information aspect of the business view. This ensures a high cohesion inside each block as well as decorrelation between blocks in order to build a modular and flexible architecture;

- **Processes aspect of the application view**

This aspect reflects the perspective of the application (i.e. software) architect and represents the application modules required to implement the functional blocks. Firstly, it is recommended to conduct an inventory of the existing softwares likely to implement the different functional blocks. Then, if none of the existing applications are able to satisfy new business process needs, the software architect chooses to set up new software components. In addition, he specifies the links between applications (exchanged messages, data synchronization, transferred files, etc.).

Besides processes and information aspects, we explicitly state goals for each view. Business goals are refined into functional goals, which are refined into application goals.

Alignment and consistency issues are critical for EA [12]. Our main contribution is to dedicate a specific view to address them: the **integration view**. This view defines an alignment map by specifying (1) entities to align (2) the alignment and consistency links (3) appropriate model transformations required to refine these entities. Model transformations ease and automate transitions from one view to another. Figure 4 provides a metamodel of our overall approach. It represents the discussed entities and their relationships. Other views and aspects can extend it.

The integration view allows for "vertical" integration (between views) and for "horizontal" integration (within one view).

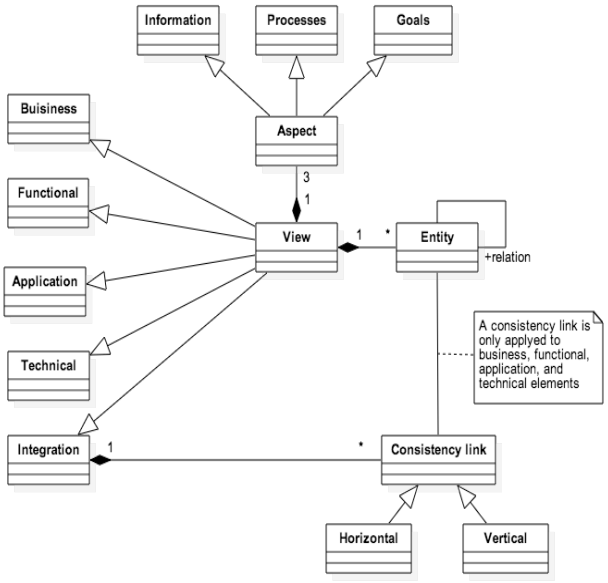

Figure 4. Metamodel of the approach

Vertical integration describes consistency links between EA views except for the integration view. Hence, the transverse integration view makes it possible to explicitly model "Refinement" links showed in Figure 3 through the class "Vertical".

We give a model of the integration view in Figure 5. For instance, it verifies that an application actually implements all functional blocks necessary for the execution of a business process. This view gives access

to traceability information in order to analyze the impact of an evolution or a failure of an application module on the business processes. It also verifies that an application format can encode functional data types representing a given business concept. Besides it identifies the potential model transformations required to automate the deployment and the integration of models from different views. Thus, model transformations are seen as operators on models and Model Typing as data typing.

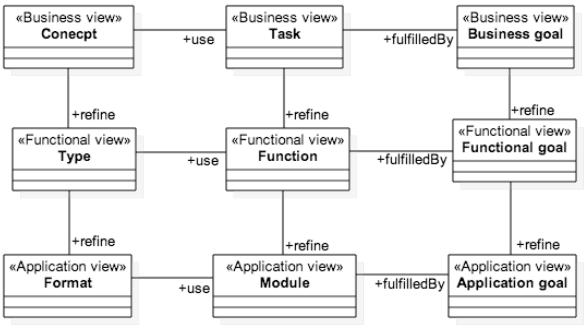

Figure 5. Entities and relationships involved in the integration view

Likewise, the integration view checks intra-consistency links through the "Horizontal" class. In Figure 5, it takes the form of the "use" association between information and processes aspects. It ensures the compatibility of data exchanged among the tasks of a business process, the functions of a functional process, and the modules of an application.

The "fulfilledBy" association of Figure 5 also inherits from the "Horizontal" class. It explicitly associates a goal and a responsible entity.

Moreover, MDE methods and languages are valuable to automate the analysis of EA. The selection of the relevant MDE techniques is guided by the IBM manifesto [6], which recommends using languages that are executable, standardized, and familiar to experts' domain. For instance, this is the case of fUML, BPMN, and OCL. Moreover, the selection of the relevant languages for model transformation depends on the involved models (textual, graphic, etc.).

4.2. Implementation of the approach for the case study

In this section, we experiment our approach on the management of an electric vehicles fleet. We create the required models for business, functional and software views adopting executable languages. Consistency links are modeled within the integration view. Figure 6 illustrates the overall architecture of the industrial case.

- **Business view**

For this view, we use fUML as an executable modeling language for business processes. Our business process consists in assigning a vehicle (electric or combustion) to a giving tour. We model and simulate this process using Papyrus, which supports fUML. Thus, the created model represents the business view processes aspect (Figure 6).

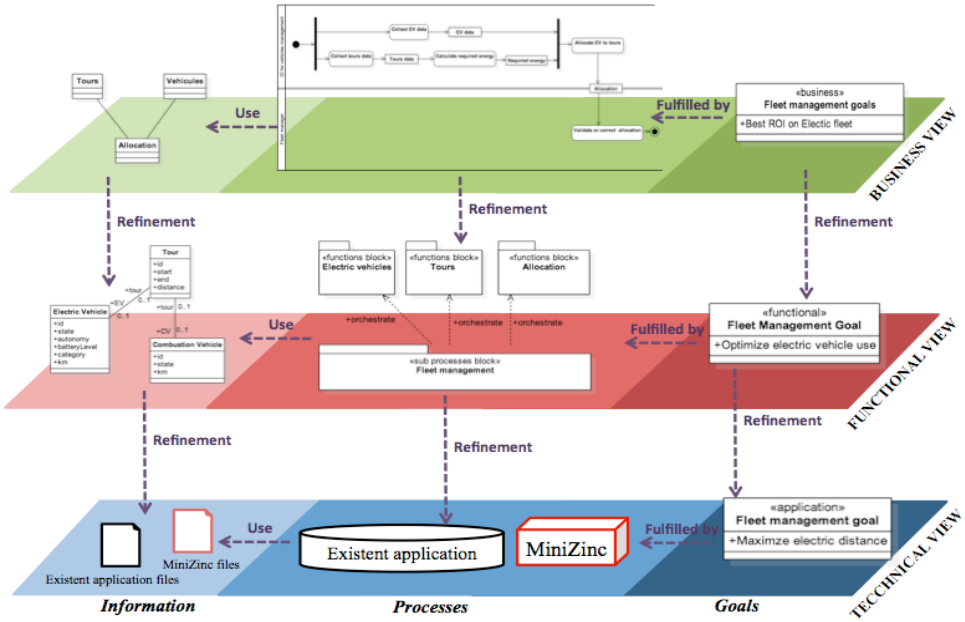

Figure 6. General architecture of the study case according to our approach

The business process begins with collecting data concerning vehicles and tours. Then it estimates the energy needed by each tour before assigning the right

vehicle to each tour. Finally, the fleet manager validates or corrects this allocation.

For the information aspect, we model the concepts of vehicle, tour and allocation using a UML class diagram that specifies the class names and their relationships.

Managing a fleet of vehicles may have various goals. For this case, getting the best ROI (Return On Investment) is a business goal for the integration of electric vehicles into the fleet. The business goal is modeled as a UML class.

Domain practices motivate the choices regarding languages and tools. Indeed, the International Electric Commission relies on UML class diagrams to model and maintain the CIM standard [24]. CIM is a Common Information Model for the electric field.

- **Functional view**

The functional processes aspect contains several blocks. One block is dedicated to functional processes that refine the business process. These processes are constituted of functions orchestration. Further, there is one block gathering together functions handling the allocation, one block for the vehicles, and one block for the tours (Figure 6).

For instance, we model the function that performs the allocation by creating a number of OCL constraints to solve (Figure 7). Indeed, in order to allocate an electric vehicle to a tour, its autonomy must be enough for the given tour. In this case sudy, we consider that there is no possibility to recharge the vehicle during the tour.

```
context Tour
inv : self.EV.autonomy * self.EV.batteryLevel > self.neededEnergy

context Tour
inv : self.EV <> undefined xor self.CV <> undefined

context Tour :: elecKm() : int
body :  (Tours::allInstances() -> collect(t.EV <> undefined  |t.distance)) ->
sum()
```

Figure 7. OCL constraints for the allocation function

We model the functional data by a UML class diagram, which represents the information aspect of the functional view. Besides, the OCL constraints are applied to this class diagram. Indeed, OCL is suitable for class diagrams. It is an executable language and an OMG standard for expressing constraints. For the other treatments (calculating tour energy, etc.), it is possible to use executable activity diagrams.

The functional goal is to optimize the use of the electrical vehicles in order to reach the business goal. Indeed, an electric vehicle has to be frequently used in order to be economically profitable. The functional goal is also modeled through a UML class as illustrated in Figure 6.

- **Application view**

For the dynamic aspect of this view, we identify the applications needed to implement the functional blocks. In our case, the application portfolio of the company already has an application for tour management (calculate optimized tour from work orders), and for vehicle management (administration, maintenance, etc.). For allocation, we use MiniZinc to model the identified constraints Figure 8.

MiniZinc is a medium-level constraint modeling language. It is high-level enough to express most constraint problems easily, but low-level enough to be mapped into existing solvers. It aims to become a standard modeling language in the field of constraint-oriented programming.

```
%a vehicle is affected to only one tour
constraint alldifferent(allocation);

%allocate an electric vehicle if the autonomy permits it
constraint forall(i in  Tours, j in ElectricVehicles)
(allocation[i]= id_EV[j] -> (autonomy[j] > neededEnergy_Tours[i]
/\ kmElec[i]= distance_Tours[i]));

% allocate a combustion vehicle
constraint forall(i in  Tours, k in CombustionVehicles)
(allocation[i]= id_CV[k] -> kmElec[i]=0);

%maximze km done by electric vehicles
solve maximize sum(i in Tours)(kmElec[i]);'
```

Figure 8. Constraints in the MiniZinc module

The different data files processed by applications represent the information aspect of the software view (Figure 9).

```
number_EV = 4;
ide_EV = [1, 2, 3, 4];
number_CV = 4;
id_CV = [11, 22, 33, 44];
autonomy_EV = [1500, 1200, 1500, 1200];
number_Tours = 4;
distance_Tours = [1000, 700, 800, 1600];
```

Figure 9. Data file for the MiniZinc module

- **Integration view**

We model the integration view by (1) the links between elements from the same view or between elements belonging to different views and (2) the constraints on these links. For instance, in Figure 10, we modeled the function “Allocate EV to tour”, its inputs and outputs, and the functional goal it fulfills. We did the same for the MiniZinc module “Optimize allocation”, for its inputs and outputs, and for the application goal it fulfills.

Due to the limitation of space, we only illustrate these elements of the overall case study. Nevertheless, the same principles are applied to the remaining elements from the different views.

For instance, an "input" link should check that the information type is consistent with the function using it and check that the information format fits the involved application module. Besides, these constraints ensure a good orchestration of tasks, functions and modules. For example, it ensures that the output of a given function is consistent with the input of the next function. The links "input", "output", and "fulfilled by" inherit from the class "Horizontal" of the approach metamodel illustrated in Figure 4.

If a function output does not fit the information type of the next function, we use Model Typing as explained in **Erreur ! Nous n'avons pas trouvé la source du renvoi.**. For instance, the function that calculates the required energy for a tour may not use a tour as information but rather a ride.

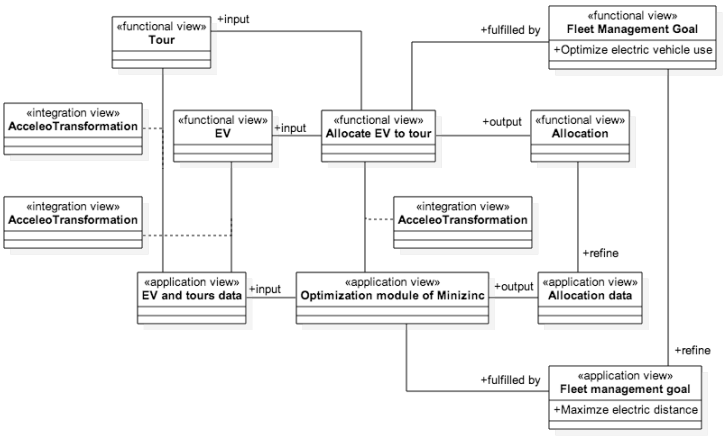

Figure 10. Partial integration view model for the EV allocation case study

In this case, we define "Tour" as subtype of "Ride" in order to perform the calculation on "Tour" and to be able to orchestrate the functions. Indeed, a function can also be seen as a model transformation that takes some models as inputs and produces other models as outputs.

The integration view also establishes links across elements of different views. In the example above, the "refine" association between the functional goal and the application goal insures Business/IT goal alignment. This kind of consistency link checks if all applications are directly or indirectly linked to the goals provided by the functional view which are linked to the goal of the business view. For instance, to *Optimize electric vehicles use*, the MiniZinc module has to *Maximize electric distance* while calculating the allocation (Figure 10).

Model transformations can also guarantee consistency between views in addition to facilitating the shift from one view to another. For instance, OCL constraints for the allocation function are automatically transformed into MiniZinc constraints in the application view using the transformation language Acceleo. We also transformed the information types "Electric Vehicle" and "Tour" into the right information format for the MiniZinc module.

5. Related Work

Model Driven Engineering has proven its capacity to address complex software systems [9]. Our message in this paper regarding MDE is twofold. First, it is suitable for EA modeling. Secondly, it is above all suitable for the automated analysis, prediction, and simulation of the whole Enterprise Architecture, not limited to the application and technical views. *Models-as-executable-artifacts* is the least common of all models usages in EA [14]. However, ontology approaches allow for automated analysis of models for specific aspects as tasks and workflows [10] or for a holistic view of EA in more recent ontology approaches [23]. Nevertheless, these approaches requires to map *models-as-high-level-specifications* [14] and their equivalent ontologies in order to perform analysis. Our approach does not require mapping and performs analysis and simulation immediately on the executable models as designed by stakeholders.

Besides, intra- and inter-view consistency, validation and impact analysis are addressed by a UML based approach where an extension of UML provides a toolkit of EA entities and their relationships [2]. Our study relies on these entities and relationships but goes further by facilitating the shift from one view to another using model transformation and sticks to the standard executable languages like fUML and OCL.
Moreover, Model Driven Architecture (MDA) applied at the level of an organization [7] deals with enterprise models as it would do with a software system. The business and function views are seen as Computation Independent Models, and the application and technical views as Platform Specific Models. This vision joins our approach on its principles regarding the need for well-defined modeling languages. However, our aim is to provide enterprise modelers with an executable Model Driven framework in line with their current practices in order to ease its adoption.

6. Conclusion and perspectives

Existing EA frameworks provide a holistic picture of enterprise systems but lack executability for modeling validation and analysis. Modeling assessment and analyzability help on addressing the key challenge of business/IT alignment. Model Driven Engineering applied to EA provides enterprise modelers with an efficient toolkit of executable modeling languages and techniques like model transformation and Model Typing. Given this statement, our contribution is to provide an executable

Model Driven framework for EA with entities and relationships across different EA views and with the relevant MDE methods. We modeled inter- and intra-view links in a dedicated view: the integration view. The integration view allows for checking the horizontal consistency between behavior, information, and goals, as well as the vertical refinements of behaviors, information and goals across views.

We apply our approach on the Smart Grids context. Indeed, we show how Smart Grids have a major impact not only on power grids but also on the whole enterprise system of grid operators.

Currently, our approach is used to create machine-processable models and conduct automated analysis regarding the As-Is or the To-Be of an EA in separate ways. However, to retrace the life cycle of EA and the location of EA in this life cycle, we need an evolution framework. Research is being conducted in this direction but at the Information System level [16]. Therefore, we are continuing to enrich our executable Model Driven framework to take into consideration the evolution aspect of EA throughout its life cycle.

7. References

[1] Armour, F.J., Kaisler, S.H., and Liu, S.Y. A big-picture look at enterprise architectures. *IT professional 1*, 1 (1999), 35–42.

[2] Armour, F., Kaisler, S., Getter, J., and Pippin, D. A UML-driven enterprise architecture case study. IEEE (2003), 10–pp.

[3] Bézivin, J. In search of a basic principle for model driven engineering. *Novatica Journal, Special Issue 5*, 2 (2004), 21–24.

[4] Bézivin, J. On the unification power of models. *Software & Systems Modeling 4*, 2 (2005), 171–188.

[5] Bézivin, J. and Gerbé, O. Towards a precise definition of the OMG/MDA framework. IEEE (2001), 273–280.

[6] Chesbrough, H. and Spohrer, J. A research manifesto for services science. *Communications of the ACM 49*, 7 (2006), 35–40.

[7] Clark, T., Kulkarni, V., Barn, B., France, R., Frank, U., and Turk, D. Towards the Model Driven Organization. IEEE (2014), 4817–4826.

[8] Favre, J.-M. Towards a basic theory to model model driven engineering. Citeseer (2004), 262–271.

[9] France, R. and Rumpe, B. Model-driven development of complex software: A research roadmap. IEEE (2007), 37–54.

[10] Fraser, J. and Tate, A. *The Enterprise Tool Set: An Open Enterprise Architecture.* AIAI, University of Edinburgh, 1995.

[11] Jézéquel, J.-M., Barais, O., and Fleurey, F. Model driven language engineering with kermeta. In *Generative and Transformational Techniques in Software Engineering III*. Springer, 2011, 201–221.

[12] Kaisler, S.H., Armour, F., and Valivullah, M. Enterprise architecting: Critical problems. IEEE (2005), 224b–224b.

[13] Kleppe, A.G. A language description is more than a metamodel. (2007).

[14] Kulkarni, V., Roychoudhury, S., Sunkle, S., Clark, T., and Barn, B. Modelling and Enterprises-The Past, the Present and the Future. (2013), 95–100.

[15] Mens, T. and Van Gorp, P. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science 152*, (2006), 125–142.

[16] Métrailler, A. and Estier, T. EVOLIS Framework: A Method to Study Information Systems Evolution Records. IEEE (2014), 3798–3807.

[17] Monti, A. and Ponci, F. Power grids of the future: Why smart means complex. IEEE (2010), 7–11.

[18] Muller, P.-A., Fleurey, F., and Jézéquel, J.-M. Weaving executability into object-oriented meta-languages. In *Model Driven Engineering Languages and Systems*. Springer, 2005, 264–278.

[19] Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., and Tack, G. Minizinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming–CP 2007*. Springer, 2007, 529–543.

[20] Palensky, P., Widl, E., and Elsheikh, A. Simulating cyber-physical energy systems: challenges, tools and methods. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on 44*, 3 (2014), 318–326.

[21] Shannon, R.E. *Systems simulation: the art and science.* Prentice-Hall Englewood Cliffs, NJ, 1975.

[22] Steel, J. and Jézéquel, J.-M. On model typing. *Software & Systems Modeling 6*, 4 (2007), 401–413.

[23] Sunkle, S., Kulkarni, V., and Roychoudhury, S. Analyzing enterprise models using enterprise architecture-based ontology. In *Model-Driven Engineering Languages and Systems*. Springer, 2013, 622–638.

[24] Uslar, M., Specht, M., Dänekas, C., et al. *Standardization in Smart Grids: Introduction to IT-Related Methodologies, Architectures and Standards.* Springer Science & Business Media, 2012.

[25] www.smartgrids-cre.fr.

[26] ec.europa.eu/programmes/horizon2020.

[27] wiki.eclipse.org/OCL/OCLinEcore.