A SysML-based methodology for mechatronic systems architectural design

Faïda Mhenni, Jean-Yves Choley, Olivia Penas, Régis Plateaux, Moncef Hammadi

A SysML-based methodology for mechatronic systems architectural design

Faïda Mhenni *, Jean-Yves Choley, Olivia Penas, Régis Plateaux, Moncef Hammadi

LISMMA-SUPMECA, 3 Rue Fernand Hainaut, Saint-Ouen, France

Mechatronic systems are characterized by the synergic interaction between their components from different technological domains. These interactions enable the system to achieve more functionalities than the sum of the functionalities of its components considered independently. Traditional design approaches are no longer adequate and there is a need for new synergic and multidisciplinary design approaches with close cooperation between specialists from different disciplines.

SysML is a general purpose multi-view language for systems modeling and is identified as a support to this work.

In this paper, a SysML-based methodology is proposed. This methodology consists of two phases: a black box analysis with an external point of view that provides a comprehensive and consistent set requirements, and a white box analysis that progressively leads to the internal architecture and behavior of the system.

1. Introduction

Over the last decades, complexity of systems has considerably grown and systems are involving more and more functionalities and new innovative technologies. New functionalities are achieved by the interconnection and synergic interactions between components from different disciplines such as mechanical, electric/electronic and software components [1,2]. Such systems, called mechatronic systems are characterized by their synergic and multi-disciplinary aspect [1] and are equipped with various components such as sensors, electronic components, actuators and mechanical structures that interact in order to perform the overall function(s) required. A successful design can be achieved only if the overall system is considered and the interactions between components are thoroughly understood and planned [1]. Traditional design for multidisciplinary systems employed a sequential design-by-discipline approach. In such approach, the design is made up of a succession of sequences. The design of an electromechanical system for instance, is often accomplished in three steps. First, the mechanical parts are designed and when the mechanical design is complete, the power and microelectronics are designed and finally, the control algorithm is designed and implemented [3]. In this sequential approach, each sequence locks some aspects of the design that become additional constraints to the next sequence. In this way, the final system is not optimized. Another drawback of this approach, is that it is time consuming since each team can begin working only when the previous has finished. Sequential approach is thus no longer adequate and there is a need for new synergic and multidisciplinary design approaches and tools [4] allowing close cooperation between specialists from different disciplines [2,5]. To cope with this need, new approaches, standards and tools emerge.

Systems Engineering (SE) is an "interdisciplinary approach and means to enable the realization of successful systems" [6] that focuses on the entire system rather than on different components independently. SE is thus appropriate for the design of complex systems [7], for instance mechatronic systems. Many standards dealing with SE arose to describe the processes and activities to be undertaken to assist the engineering of complex systems. The IEEE 1220 standard [8] focused on the technical processes of SE from the requirements analysis to the definition of the physical system. Three processes are described in detail in this standard: Requirements analysis, Functional analysis and allocation and, Synthesis, each process having verification and validation sub-processes. The EIA 632 standard [9], covers a larger scope since, in addition to the technical processes of system design, it includes the acquisition and supply, technical management, product realization and technical evaluation processes. Finally, the third well known standard about SE, the ISO-IEC-15288 [10], describes the processes of the whole lifecycle of a product.

* Corresponding author. Tel.: +33 149452537.
E-mail address: faida.mhenni@supmeca.fr (F. Mhenni).

In addition to the process standards cited above, some tools and languages were developed in order to support the processes described in the standards. Among systems engineering dedicated languages, SysML [11,12], the object-oriented Systems Modeling Language developed by the Object Management Group (OMG) and the International Council on Systems Engineering (INCOSE) is being widely used for various purposes in both academic and industrial fields [13–18].

Having experienced SysML in the O2M (Modeling and Design Tools for Mechatronics) French research project [19] as well as in different teachings, the authors are convinced of its capabilities in the design process in general and especially in the mechatronics design. These capabilities are discussed along this paper.

However, the availability of standards and tools is not enough for the successful development of complex systems. Only an appropriate methodology can lead to an optimal use of the available tools by providing a set of practices, procedures and rules to follow during the design phase. As long as we know, few methodologies were developed to describe how to achieve the appropriate processes of the available standards with SysML. This led the authors to developing their own methodology using SysML trying to take benefit of this language and the model based approach, be compliant with the standards processes while not loosing the key concepts they have been using for several years like APTE-FAST-SADT based methodologies. These methodologies, although relevant, are not implemented by tools allowing to build a consistent complex (multi-view, multi-level) model. These concepts were enriched and expressed with some of the SysML diagrams presented in the proposed methodology.

This methodology, based on the authors' experience in mechanical engineering design and on the knowledge acquired from their contribution in several research projects as part of multi-disciplinary teams, is presented in this paper.

The paper is organized as follows. A brief summary of the most known SE methodologies is given in Section 2. Section 3 discusses the adequacy of SysML for mechatronic systems design. The proposed methodology is given in Section 4 through a case study example. A discussion is given in Section 5 and finally a conclusion is given in Section 6.

2. State of the art

The most widely used methods for systems engineering are summarized in [20] and include Harmony-SE, OOSEM (Object-Oriented Systems Engineering Method), RUP-SE (Rational Unified Process for Systems Engineering) and Vitech Model-Based System Engineering (MBSE) Methodology. Among these methods, the two more popular methodologies Harmony-SE and OOSEM are described hereafter.

Rational Harmony for Systems Engineering (Harmony-SE).

Harmony-SE is part of an Integrated Systems and Software Development Process: Harmony. It is a service request driven approach using SysML language ([20,21]). It includes three top-level processes: *Requirements analysis* process consisting in collecting stakeholder requirements and translating them into (functional and non-functional) system requirements, *System functional analysis* process consisting in translating functional requirements into a coherent representation of system operations and finally *Design synthesis* process consisting in the development of a system architecture satisfying both functional requirements and performance constraints. Most of the artifacts of this methodology are built with SysML diagrams.

Object-Oriented Systems Engineering Method (OOSEM).

OOSEM is a "top-down, scenario-driven process that uses SysML (the SysML specification was adopted by OOSEM since 2006) to support the analysis, specification, design, and verification of systems" [22]. OOSEM includes the following development activities [20] *Analyze stakeholder needs*, *Define system requirements*, *Define logical architecture*, *Synthesize candidate allocated architectures*, *Optimize and evaluate alternatives* and *Validate and verify system*. These activities are performed in an iterative way and can be applied at each hierarchy level of the system. Management process is used to support each of these activities. SysML is the predominant modeling language used in generation of the activities artifacts.

In these methodologies, system functionality is mainly scenario-based or service request-based and serves as a support for the software coding. There is a lack of a hierarchical breakdown of system functions that facilitates the definition of alternative architectures by means of different functional groupings. There's also no traceability links clearly established between system functions and components in the presented methodologies. The contribution of this paper is to provide a methodology that copes with these lacks.

Indeed, having a mechanical engineering background, the authors are used to functional analysis tools like SADT (Structured Analysis and Design Technique) also known as IDEF0 (Integration Definition for Functional Modeling) [23] that allows a successive breakdown of system functions into different hierarchical levels also showing the flows distribution. This way of functional analysis gives a better understanding of system functionality and allows innovation through the different possible ways of grouping the system functions and allocating them to components. Different system architectures can then be defined and compared. The extensions included in SysML to support systems modeling include these constructs (see Section 3.2.2) that are maintained in the proposed methodology.

3. SysML for mechatronic systems design

In this section, the adequacy of SysML for mechatronic systems design will be discussed. For this purpose, Section 3.1 first describes the requirements for a successful design of mechatronic systems. Then Section 3.2 gives an overview of SysML diagrams and how far they match with mechatronic systems design modeling requirements.

3.1. Mechatronic system design

Mechatronic systems are multi-disciplinary systems and contain subsystems or components from different integrated disciplines interacting together. Their design aims at satisfying system-level requirements or functionalities. These are only obtained thanks to the interactions between components. The old fashion of splitting design into separate disciplines is no longer relevant for mechatronic design where the disciplines are interdependent. Collaborators from different disciplines have to work together simultaneously on a system-level model to understand the whole behavior of the system and the dependencies among their different disciplines [2]. This usually leads to some issues, mainly communication and understanding issues, due to their respective different backgrounds and jargon.

Building a system-level model, in a unified unambiguous language, is a key point for a successful mechatronic design. System-level model is the reference each contributor during the system design should refer, to at anytime, to have unique, up-to-date data that is shared by everyone. However, it is necessary that everybody can easily find the information needed. Then, a multi-view model with different representations of the main system is

necessary. These different views shall be linked together with traceability links and be consistent with each other.

3.2. SysML overview

SysML is a profile of UML (Unified Modeling Language) dedicated to systems engineering modeling. It has been built from the experience gathered on UML2.0 and on successful engineering graphical representations [24] such as Enhanced Functional Flow Block Diagram (EFFBD). UML was firstly used for software development and consequently it has overly software-oriented semantics. So it was not really appropriate for complex systems design namely because it does not propose a clear expression of physical flows passing between the systems components. Managing some system aspects needed proper stereotypes to be developed. These lacks led the OMG (Object Management Group) to derive a new profile specified for SE, the SysML language. It was constructed as a subset of UML, completed by additional modeling possibilities specific for SE.

SysML is designed to provide simple but powerful constructs to model a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior and allocations, and constraints on system properties to support engineering analysis [12]. It has inherited UML abilities, notably the advantages of object-oriented methods for complex system design and of its standardized aspect.

This new profile, SysML, helps performing several design tasks such as reliability study [16], complex and embedded systems modeling [25]. It is also more and more adapted by industrials such as Valeo, a supplier of automobile systems [14,26] and Airbus, a designer and integrator of aeronautic systems [13] to cite only a few examples of SysML deployment.

3.2.1. Requirements modeling with SysML

Requirement managing includes requirement capture, and then requirements analysis (defining critical requirements, making trade-offs between conflicting requirements etc.) and finally allocation (allocating requirements to use cases, components, test cases etc.). **Requirement diagram**, helps capturing, analyzing and maintaining traceability of requirements in the system modeling. A non-ambiguous expression of requirements and an appropriate allocation of requirements to corresponding parts of the system are compulsory to ensure the validation of final product.

When viewing a large number of requirements on a requirement diagrams it is very hard to depict and relate requirements. In this case, requirements can also be displayed in tables or matrices showing the desired properties and relationships among requirements and other model elements.

SysML also allows specification of test cases and linking them to the corresponding requirements to specify how these requirements should be verified.

3.2.2. Behavior modeling with SysML

SysML provides different diagrams to model different behavior kinds and thus offers a comprehensive description of behavior that helps to reach a complete specification of system [22].

- Flow-based behavior: behavior is detailed in terms of the flow of inputs, outputs and control. An **Activity diagram** represents a controlled sequence of actions that transforms inputs into outputs. It details functional architecture of the system and describes the different functions to be performed by the system and their hierarchical breakdown showing flow distribution.
- Event-based behavior is expressed in terms of response of blocks to internal and external events. State machine diagrams are used to identify different states (or operating modes) of system. Establishing such diagrams helps to define the transition from one state to another. Each mode is then treated separately and a comprehensive description of the behavior of system is established. If particular functions or requirements are needed for one state they will not be omitted. **State Machine Diagram** illustrates the different states (operating modes) the system will have through its lifecycle.
- Functionality of system, in terms of the services it provides to potential users, is represented by **use cases**. **Use Case Diagrams** represent functionalities in terms of how a system or other entity is used by external entities (such as actors) to accomplish a set of goals.
- Message-based behavior: used to model service-oriented concepts. **Sequence diagrams** represent interactions either between system and its environment or between different components of system at various levels of system hierarchy. It illustrates scenarios of use cases.

3.2.3. Structure modeling with SysML

- **Block Definition Diagram** represents structural elements that are called blocks with their properties, relationships and composition.
- **Internal Block Diagram** represents the interfaces and connections between parts of a block, it is a modification of the UML composite structure diagram.
- **Parametric Diagram** represents constraints on property values and parameters relationships. A parametric diagram aims at identifying the main system parameters as well as their relationships seen as constraints (e.g. through equations), in order to be exported to other simulation tool to be solved.
- **Package Diagram** represents the organization of a model in terms of packages that contain modeling elements.

4. SysML based methodology for mechatronic systems architectural design

4.1. Methodology overview

The development of a new mechatronic system (product) goes through different steps from requirements to the final product. These steps can be represented by the cycle in Fig. 1 [27]. As the chart shows, the development process begins with a system design phase. In this phase, the system is considered as a whole. The system level specifications and constraints are identified and then derived into component level specifications while taking into account the upper level constraints. Domain specific design is then started in a coordinated way such as the interactions between different domains and their consequences are thoroughly accounted for as it was stated in the system design phase. The scope of the SysML-based methodology presented in this paper is limited to the *System Design* phase. Indeed, SysML is meant to complement the already existing domain specific tools by bringing a system view which the domain specific views shall keep consistent with. As a complement to the methodology presented in this paper, domain specific design is performed with a variety of other tools such as Computer Aided Design and Finite Element Analysis tools.

The proposed SysML-based methodology aims at assisting the designer in the system design phase to have a consistent modeling. It also attempts to guide the designer facing the variety of diagrams in SysML by giving a sequence of uses of these diagrams in different design stages. It is a two-phase modeling process. In the first phase, the system is considered as a black box and the phase, called *black-box* analysis only gives an external point of view of the system. The second phase, called *white-box* analysis is an internal point of view of the system where the internal

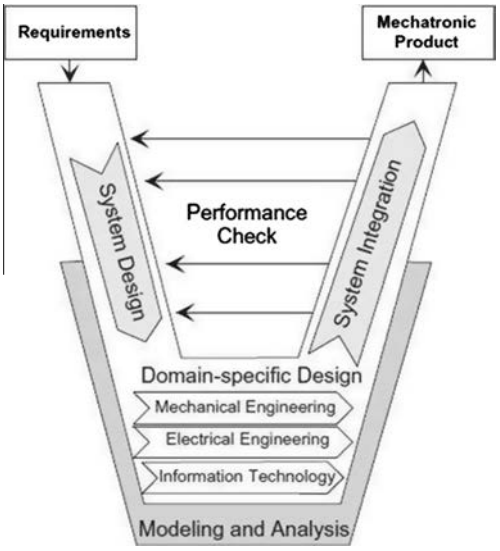


Fig. 1. Mechatronic system development cycle [27].

architecture is progressively determined. Each analysis phase is made up of several steps like described in Fig. 2. In each step, one or more SysML diagrams are used to describe a specific point of view of the system. The sequence of diagrams contributes progressively in the emergence of a consistent set of requirements for the first phase and system architectures for the second one.

Bad requirements specification could lead to significant cost and schedule overruns, failures to deliver all of the functionality specified, and systems that do not have adequate quality [28]. The aim of the *black-box* analysis is to build a comprehensive and consistent set of requirements to minimize expansive design changes due to bad specification. This is made through a sequence of steps with potential iterations between them. First, the system mission and objectives are determined. Then, the whole lifecycle is identified in order to take into account the constraints of each phase of the lifecycle. The system boundary specifying what is inside the system shall also be thoroughly identified since the early stages in the design process. For each phase of the system lifecycle, the system context including the interactions the system has with its surrounding shall be considered. Based on this, the external interfaces supporting these interactions shall be defined. Then, the external behavior (with regard to the user) of the system is modeled through the user operating modes, the services (or use cases) of the system and functional scenarios.

Requirements are collected during all the steps of the *black-box* analysis and traced to the model elements that helped in their identification and specification.

In the *white-box* analysis, the system architecture is progressively identified. First a hierarchical model of the system functions

Fig. 3. Case study the go-to-GPS telescope.

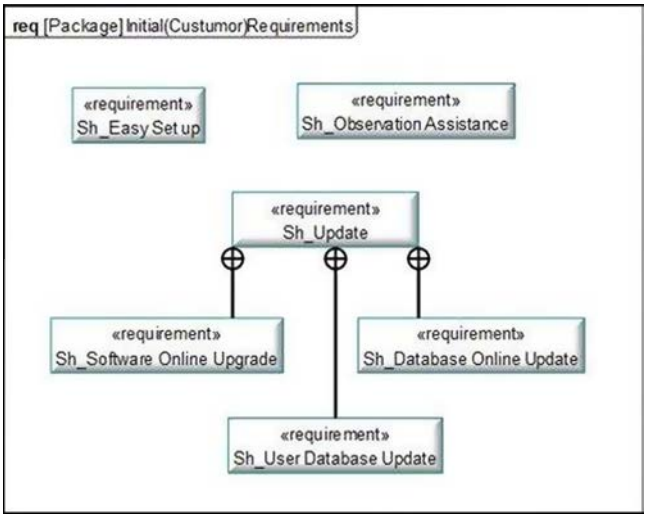


Fig. 4. Initial requirements.

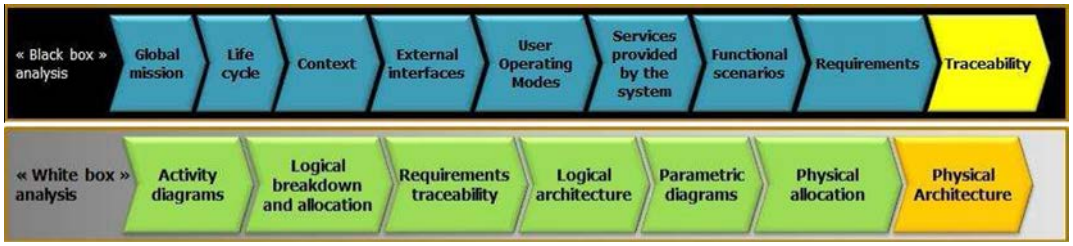


Fig. 2. Black-box and white-box analysis steps.

breakdown is established. Then, based on the functional model, components are allocated to functions to synthesize candidate logical architectures. By logical architecture we mean an architecture based on general classes of components and not specific fully defined components. Based on trade-offs and further simulations with external tools, physical architecture is defined by allocating the optimal physical (existing COTS or new) components to the logical ones. During this phase, new requirements may emerge and shall be traced to the black-box requirements and to the related model elements.

More detail about all these steps and how they are represented with SysML will be given below in the case study section. Even though these steps are given in a sequential way, in each step we can go back to previous steps either because a requirement is modified (which is a current event in industrial projects) or because the designer reminds aspects that have not be accounted for.

Thus the proposed methodology addresses the following problems. First, it gives a kind of road-map to help persons intending to use SysML choosing the right diagrams (among the variety of available diagrams) for the right purpose. It also highlights how to build a consistent model an maintain consistency between the different views of the system during all the design phase. Finally, the solution is defined progressively based on a comprehensive set of requirements thouroughly determined and a hierarchical functional modeling.

The methodology will be illustrated on a telescope named "Go-To GPS Telescope" (Fig. 3). In the next section, the case study is briefly introduced. Then, the steps of the Black-box analysis phase

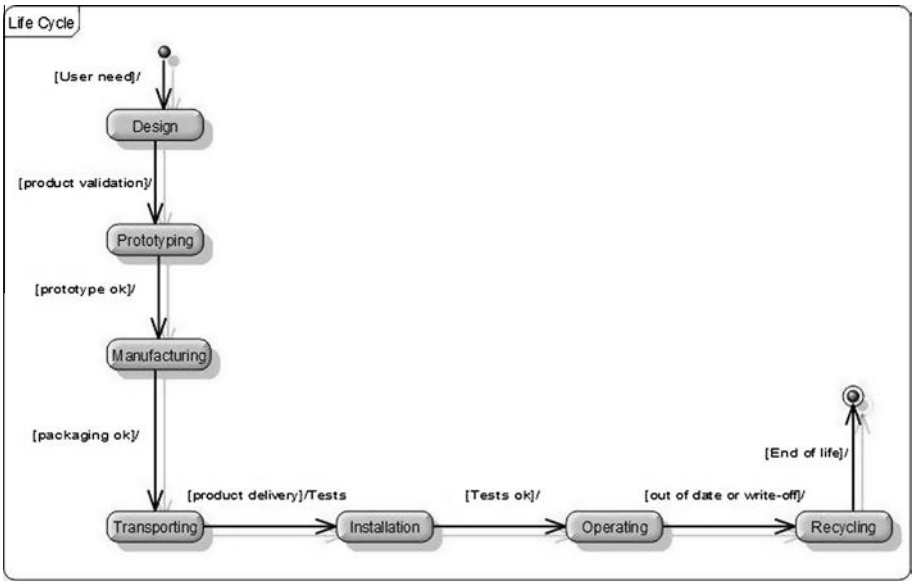


Fig. 5. System lifecycle.

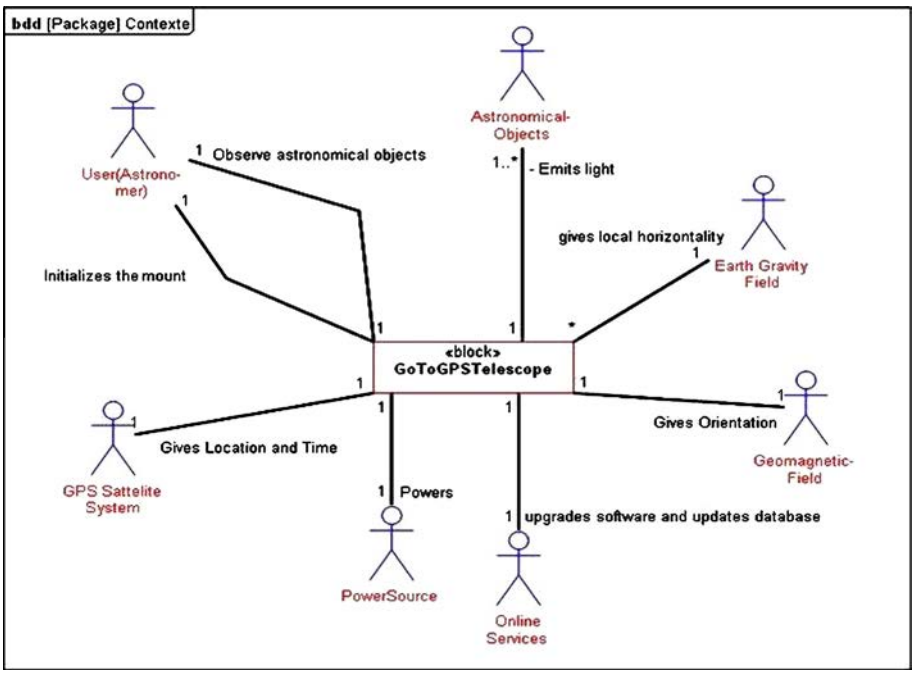


Fig. 6. Telescope context.

and the white-box analysis phase are presented respectively in Sections 4.3 and 4.4.

4.2. Case study: the go-to GPS telescope

The "Go-To GPS Telescope" used as a case study in this paper is computerized and includes a double-axis motorized half-fork mount. It is equipped with an optional GPS receiver, a built-in magnetic north sensor (compass) and an integrated electronic level sensor, in order to permit an automatic initialization of the mount and an easy access to sky objects (Fig. 3).

This Go-To technology is patented and mainly proposed by Celestron (NexStar™, Skyalign™, All-Star Polar Alignment™) and Meade (ETX, Autostar, LightSwitch™). Three operating phases will be studied: the set-up of the mount (polar alignment) in equatorial or alt-azimuth mode, the online flash upgrading of the embedded software and online updating of the built-in sky objects database, and finally the visual observation of sky objects.

4.3. Black-box analysis

The proposed methodology is a top-down approach. It begins with a black-box analysis where the system of interest is considered as a black box, meaning that its internal structure is still not defined. This modeling phase is an external point of view of the system aiming at defining a consistent set of requirements the system must satisfy, and that will be the baseline for the next phases.

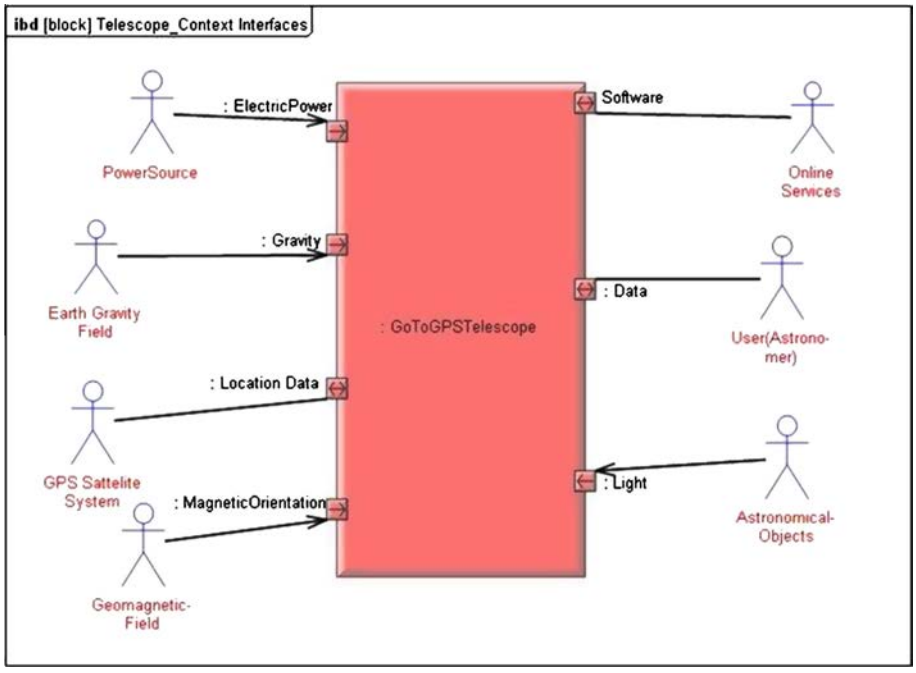


Fig. 7. External interfaces.

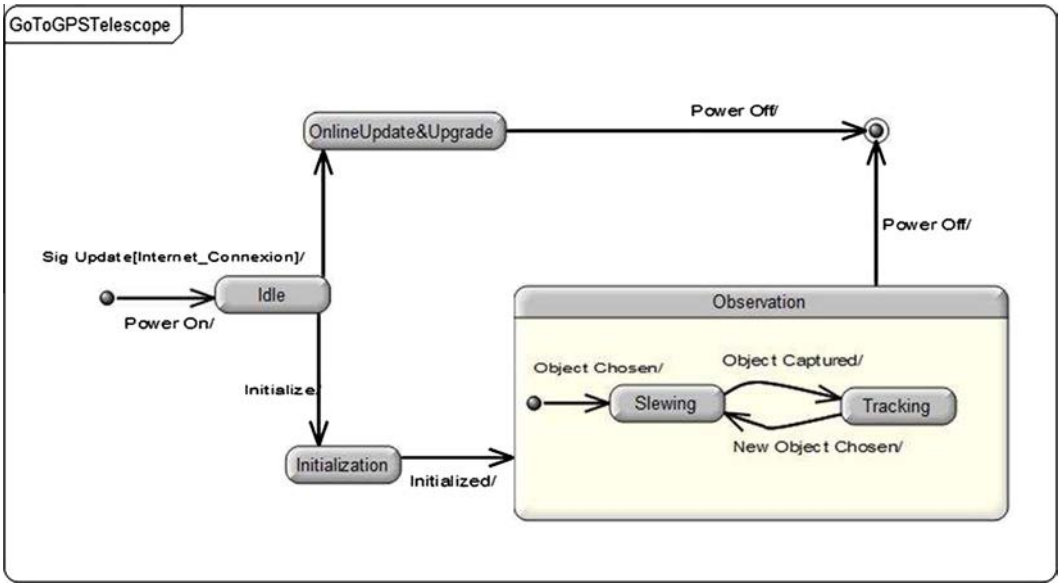


Fig. 8. User operating modes.

4.3.1. Step 1: Definition of the global mission of the system

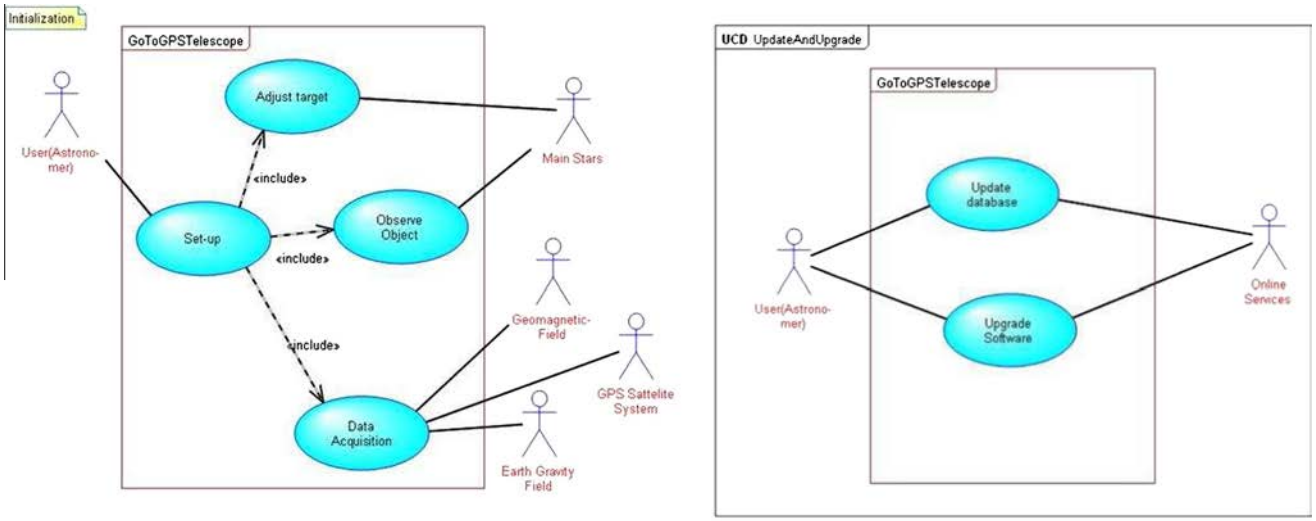
In this step, the global mission (or function) of the system shall be identified. It is usually issued from textual, oral or partial definitions of the user needs. The main mission may be given in a hierarchical decomposition into sub-functions. It is captured in SysML as one or more requirements with possible sub-requirements. A Requirement diagram (Req) represents these initial requirement(s) and their relationships to each other. Containment links are used to link the main requirement to its sub-requirements (Fig. 4). During the next steps of analysis, other requirements will be identified and will progressively be captured in the system model.

4.3.2. Step 2: Identifying the system lifecycle

In order to have a comprehensive set of requirements, all the system lifecycle phases as well as the stakeholders in each phase shall be identified, and the corresponding requirements captured. Indeed, government regulations are imposing more restrictions on all the lifecycle phases of system to take into account environmental aspects, safety measures etc. Available manufacturing, assembly, testing, transportation and all other needed means during the whole system lifecycle shall be considered. A State Machine Diagram (STM) is used in order to define the different stages of the whole lifecycle of the system and the transition conditions from one stage to another. Each stage of the lifecycle is represented by a "state". An example of lifecycle is given in Fig. 5. Each phase of this lifecycle will then be detailed in further diagrams, all along the different steps of this modeling process.

4.3.3. Step 3: Modeling the system context

It is very important to define the boundary of the system in the beginning of the study to identify what is within the boundary and what is outside, in other words, what exactly has to be developed. Boundary definition prevents from doing extra work by developing things outside the boundary, not delivering what is actually needed by excluding some parts. Then, for each phase of the lifecycle, an exhaustive list of the external interfacing elements



(a) Initialization mode UCD

(b) Update mode UCD

Fig. 9. Use case diagrams.

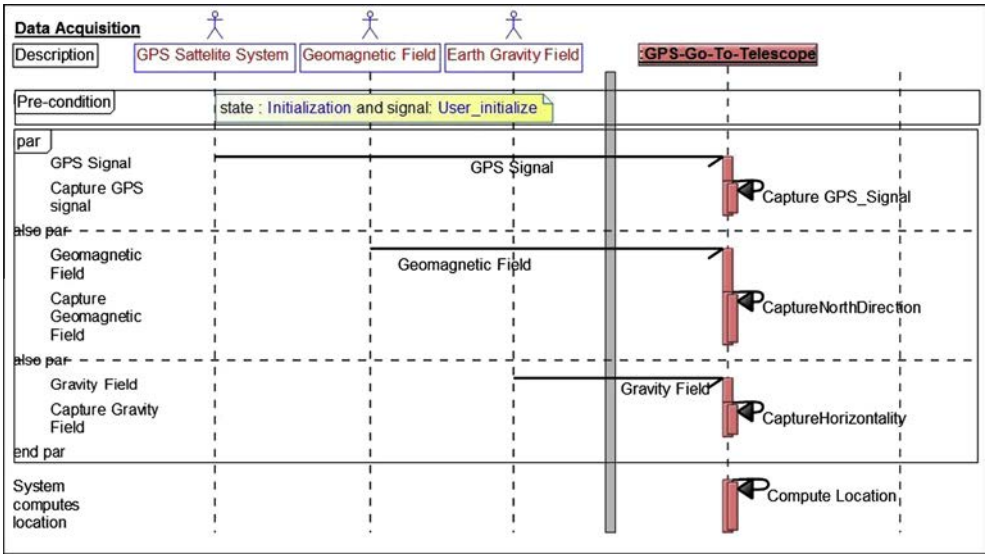


Fig. 10. The functional scenarios.

(stakeholders, external systems and actors) and their interactions with the studied system are specified. A SysML Block Definition Diagram (BDD) represents the system and its interactions with its surroundings. These interactions shall be considered during the design phase and specified within the requirements. Let's note that, in SysML, an actor (represented by the stickman icon) specifies a role played by an external entity that interacts with the subject. This external entity is not necessarily a human. To ensure consistency, each context diagram is linked to the appropriate phase of the lifecycle, i.e. to the state representing this lifecycle stage (see Fig. 6).

4.3.4. Step 4: The external interfaces

In order to define more precisely the interfaces between the system of interest and the actors that interact with it, an internal block diagram (IBD) is created to complete the BDD of the context (Fig. 7). The external interfaces shall be consistent with the context diagram i.e. all the actors that figure in the context diagram shall also be on the IBD representing the external interfaces.

Identifying the interface ports at this early stage prevents from forgetting some of them later when identifying the internal structure cause once defined, the ports are kept in the model and automatically populated in future diagrams. Consistency is then ensured both with previous steps in the black-box phase as well as with the next white-box phase. Future changes of these ports will also be automatically propagated back to the black-box models.

4.3.5. Step 5: The user operating modes

For each phase of the lifecycle (at least for the main operating phase), the user operating modes have to be determined and interconnected with a State Machine Diagram (STM) in terms of states and transitions. The Operating modes of the Telescope are represented in Fig. 8. The system operating modes detail the usage of the system during its "Operating" stage of its lifecycle. The "operating modes" state diagram is then linked to the "Operating" state of the lifecycle.

4.3.6. Step 6: The services provided by the system

For each user operating mode, each services provided by the system to the end-user is modeled by a Use Case. The dependency relationships among use cases as well as the actors that are involved in each use cas are represented in a Use Case Diagram (UCD). To ensure consistency with the system operating modes, each use case is linked with the relevant operating mode (i.e. state).

UCD diagrams for *Initialization* and *Update* user modes are given in Fig. 9. In the first diagram, we see that the Set-up use case includes three use cases. This dependency should be accounted for when detailing the system behavior.

4.3.7. Step 7: The functional scenarios

For each service (or use case), a sequential description of functional scenarios may be defined with one or more sequence diagrams (SEQ). Sequence diagrams are strongly coupled to the UC they detail. In a sequence diagram, the interactions between the system and its context (external actors and/or systems) are detailed. The same actors that are linked to the use case shall be on the corresponding sequence diagram. Internal operations may also emerge from this diagram and will be used in the functional model during the white-box analysis.

An Example of sequence diagram is given for the telescope (Fig. 10). At the right hand side we can find the lifeline of the system, and in the left hand side the lifelines of the different actors. Interactions are represented as exchanged messages (arrows on the diagram). Reflexive messages (arrows with the same source and target) represent the operations that the system shall perform.

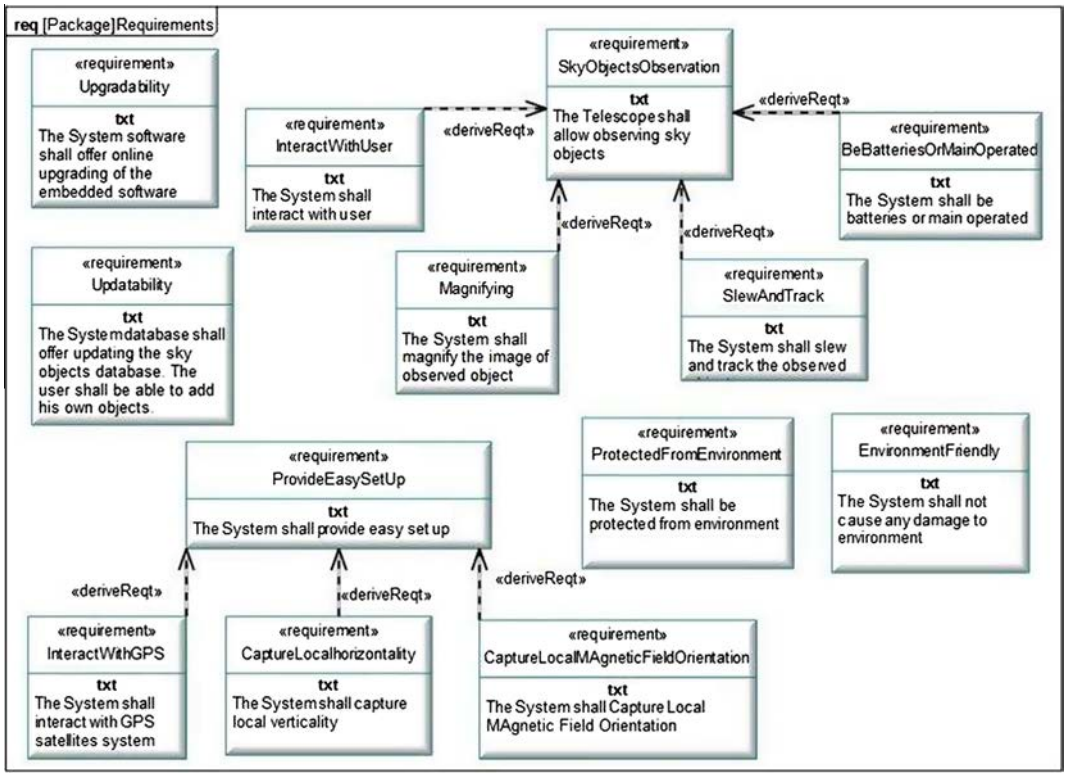


Fig. 11. Requirements specification.

4.3.8. Step 8: Requirements specification

All the information captured within the previous diagrams shall be elicited in the model database as requirements. The set of requirements has to be structured with the appropriate links among: *derive*, *refine* and *contain* relationships. These relationships can be displayed in a requirement diagram that summarizes the initial requirements (representing the global mission) as well as the requirements derived from the other diagrams of the black-box analysis like the UCD for the system services or the interactions of the BDD context. The requirements for the Telescope are given in (Fig. 11).

4.3.9. Step 9: Requirements traceability

In order to be able to trace all requirements, and ensure consistency of the black-box phase, some relationships are specified in some additional requirements diagrams (Fig. 12). The following traceability links are used:

- *"refine"* between Requirements and UC;
- *"allocate"* between Requirements and roles in BDD context;
- *"satisfy"* between Requirements and external ports (IBD).

4.4. White-box analysis

In the previous phase, an external point of view analysis was established in order to identify a comprehensive set of requirements and specifications of the system. A baseline is now available to go further and identify different candidate solutions. This modeling phase is conducted with an internal point of view on the studied system, in order to model its internal structure and behavior, with respect to the set of requirements (REQ) specified during the black-box analysis.

4.4.1. Step 10: Functional architecture

In this step, the functional requirements identified in the black-box analysis (from use cases and operations identified on

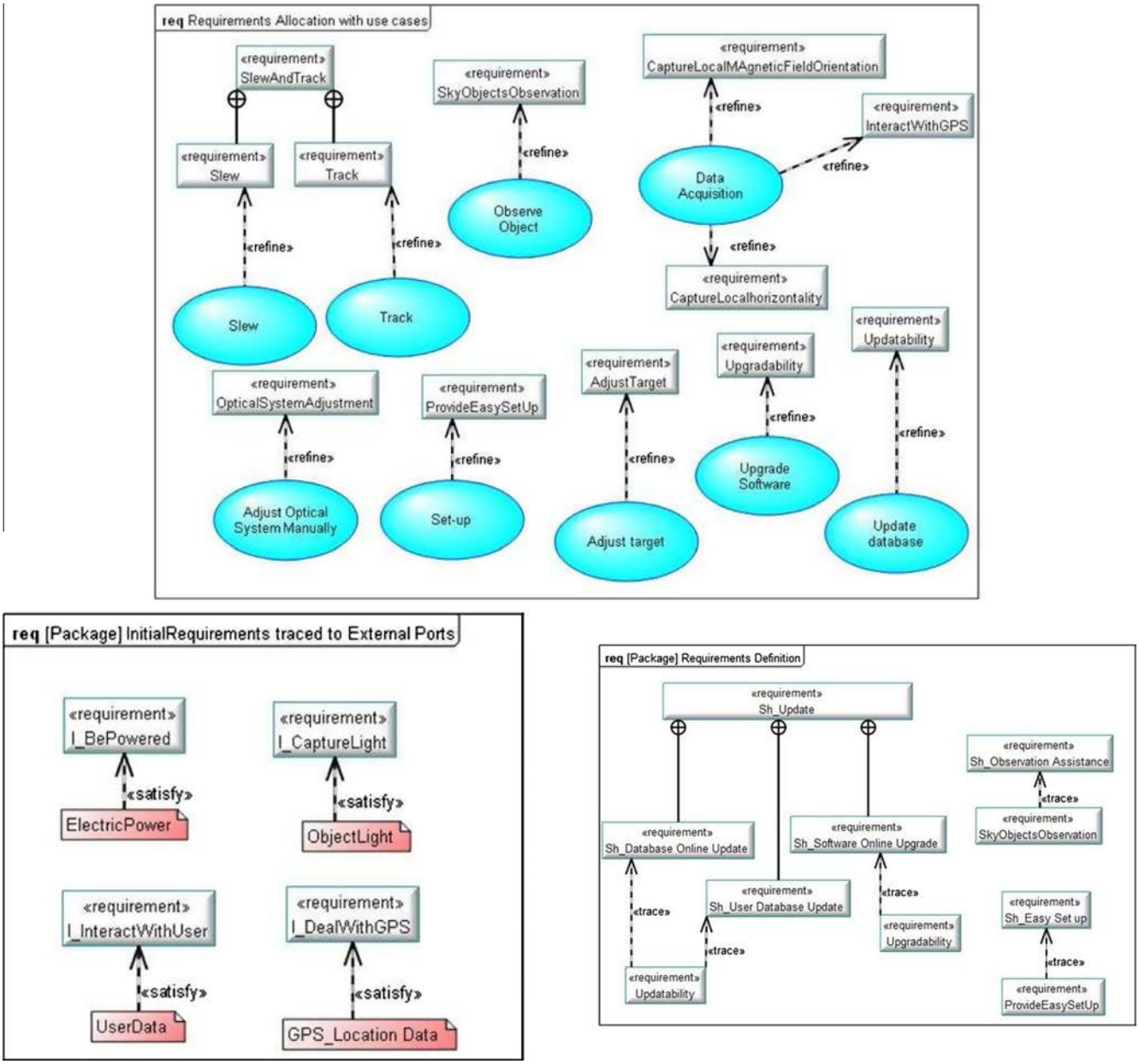


Fig. 12. The requirements traceability.

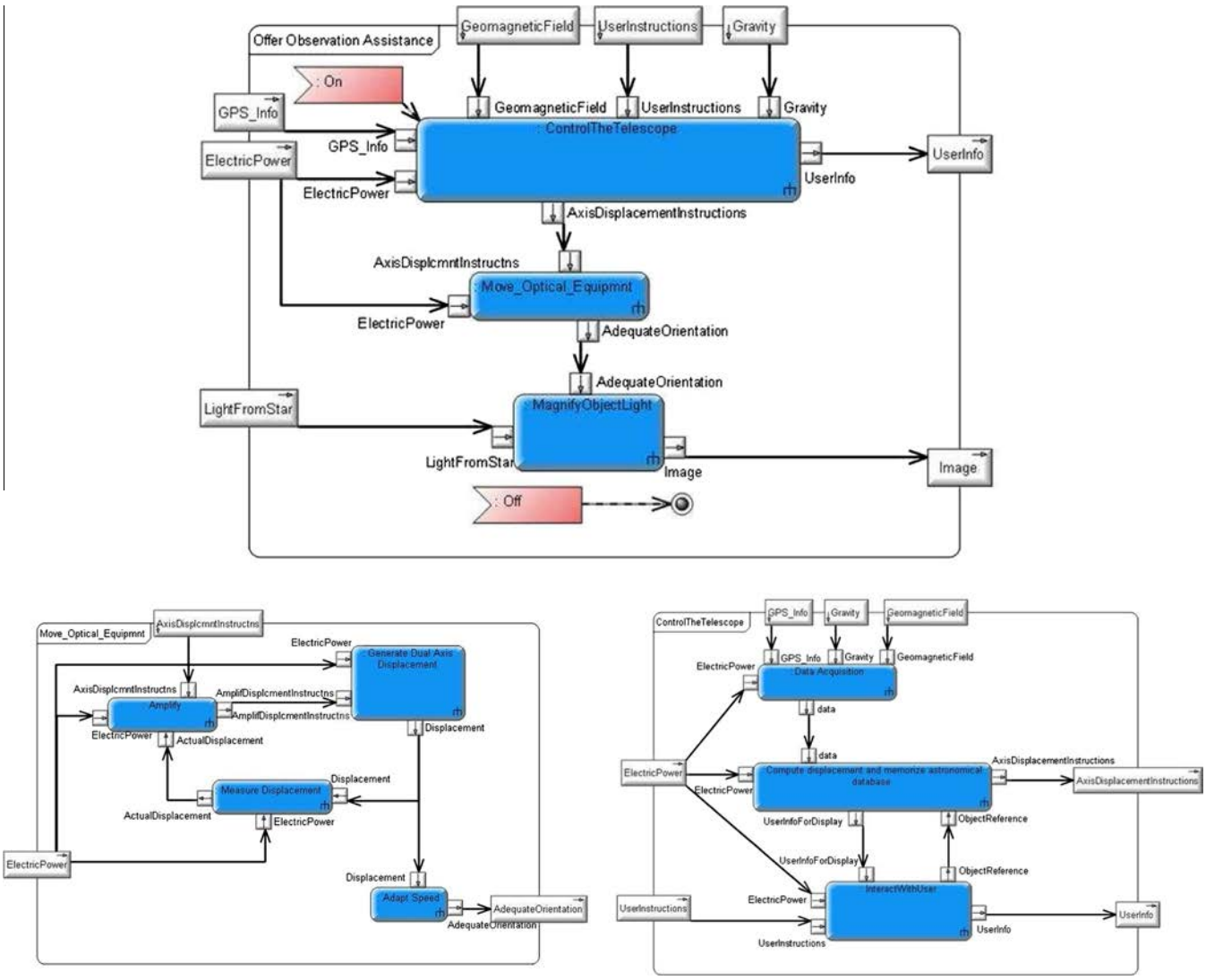


Fig. 13. The functional architecture.

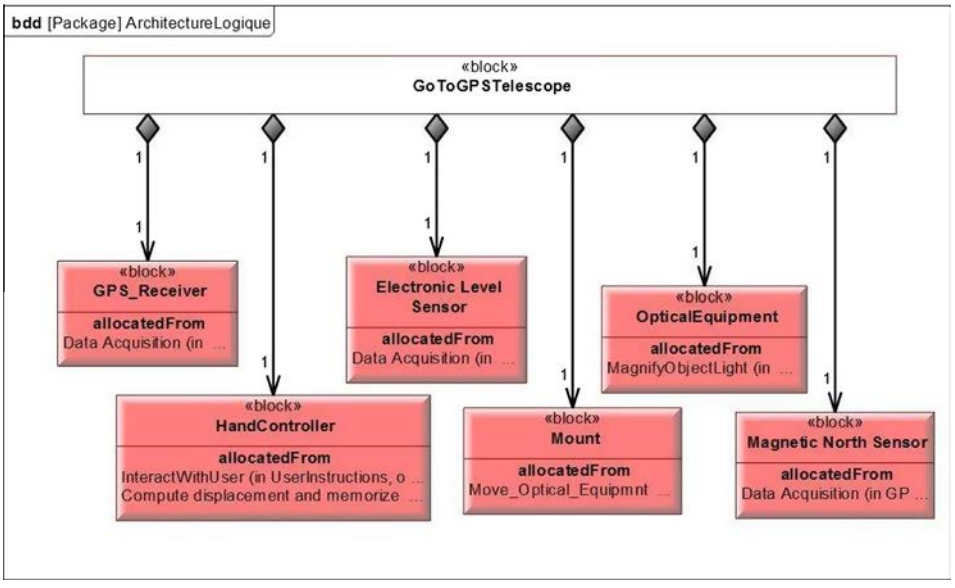


Fig. 14. Logical breakdown.

sequence diagrams) are translated into a coherent description giving the functional architecture of the system. Systems functions are represented by means of activities, and shall be linked to the corresponding requirements. The functional architecture is defined with activity diagrams (ACT), with a top-down breakdown of the main function of the system into sub-functions. Activity diagrams show the progressive transformations of input flows into output flows. Both object flows and control flows are supported. External PINs must be consistent with the external ports of the previous external interfaces IBD (step 4) (see Fig. 13).

4.4.2. Step 11: Logical breakdown and allocation (logical structure)

In this step, system functions are allocated to logical components. By logical component we mean a general class of components, mainly a kind of technology: motors, gears, etc. Thus, a set of logical components is chosen in order to achieve all functions specified in the functional breakdown (step 10). Allocations between activities and components can be displayed in a Block Definition Diagram (BDD) in Fig. 14. In this diagram, one or more functions (activities) are allocated to one logical component. At this step, several candidate logical structures can be proposed and then compared. Moreover, and to ensure consistency with black-box analysis, the operations identified in sequence diagrams can be allocated to relevant components.

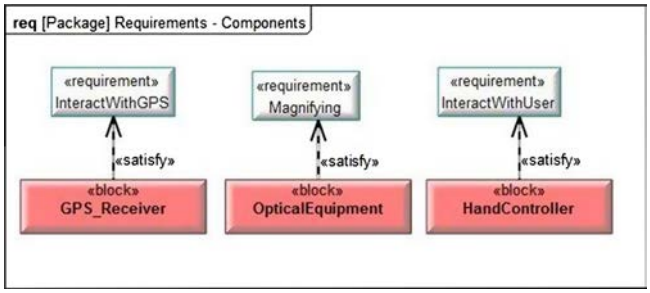


Fig. 15. Requirements to logical components traceability.

4.4.3. Step 12: Requirements to logical components traceability

Once the logical components are identified and allocated to activities, component-level requirements are derived from system-level requirements and allocated to the corresponding requirements. The logical components shall satisfy these requirements. *Satisfy* relationships are used to show this dependency and are displayed in a new requirements diagram (REQ) (Fig. 15). New requirements can also emerge from the choice of the logical components and shall then be added to the requirements database and linked to the corresponding components.

4.4.4. Step 13: The logical architecture

The logical components are now identified, but the way they interact is still not specified. In this step, the internal interactions between the components are given. An internal block diagram (IBD) displays the system architecture with the interactions among components and the different flows they interchange with each other (Fig. 16).

4.4.5. Step 14: Use of the parametric diagram

As long as we have identified a set of parameters that can be linked together with mathematical equations to describe the required behavior, and that a resolution is needed either for dimensioning or for trade-off, a parametric diagram is very useful. This can be useful for example if there are different candidate logical architectures. These could be compared in preliminary design with respect to simulation results that can be obtained with parametric diagrams.

In a parametric diagram (PAR), modeling parameters (component and system level parameters) can be linked to each other meaning constraint blocks carrying the set of equations to be solved (Fig. 17). Parametric diagram gives a description of parameters dependency in the system level model, and is intended to be simulated using external tools. It's a kind of bridge between SysML and simulation tools and allows to share the same parameters between SysML and other tools used along the design process. It also offers the possibility to feed back the system model with simulation results. It may be used at any step of the modeling process whenever equations are available and need to be solved, for

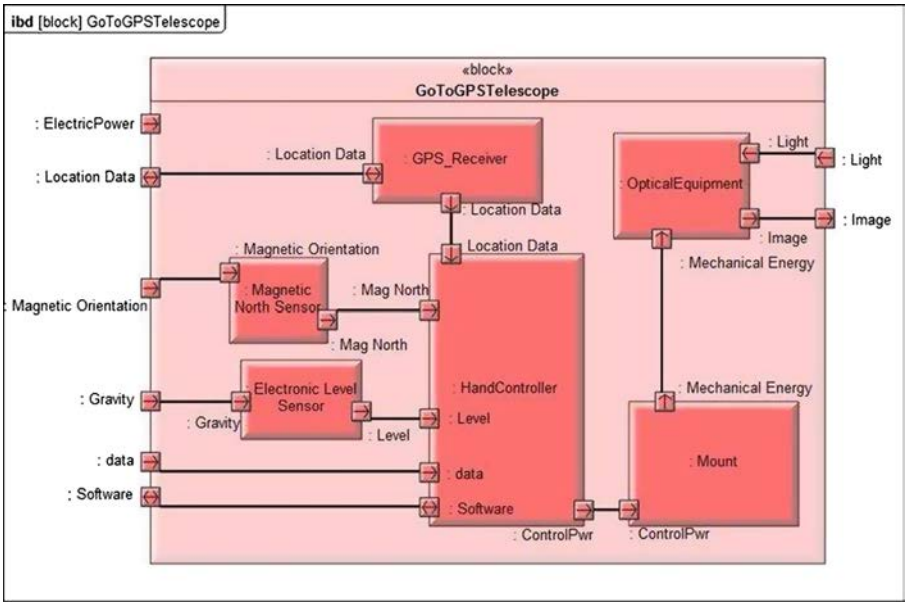


Fig. 16. The logical architecture.

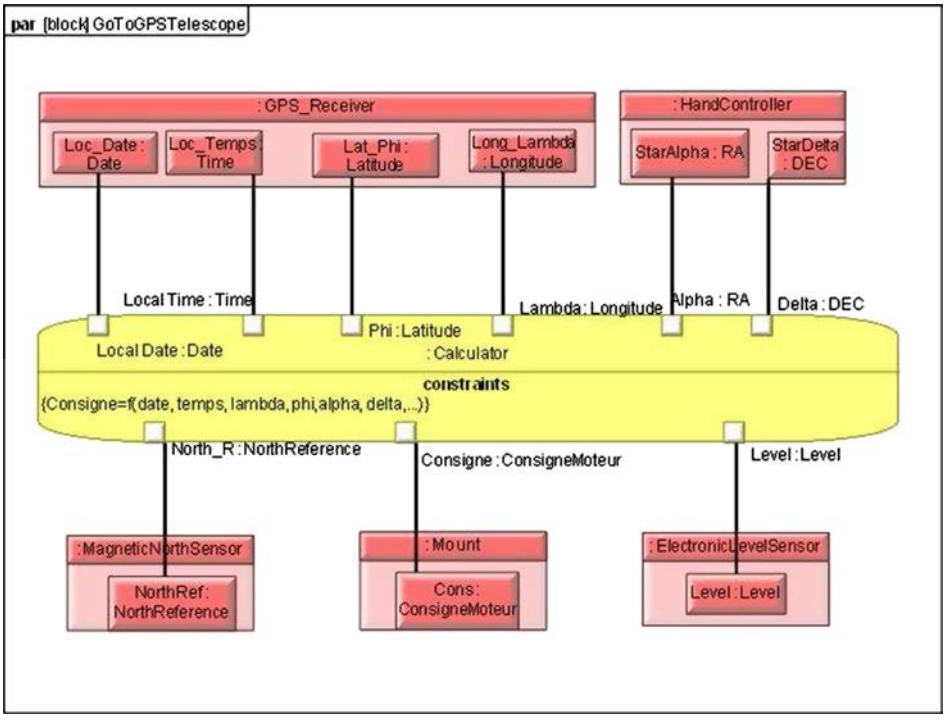


Fig. 17. Use of the parametric diagram.

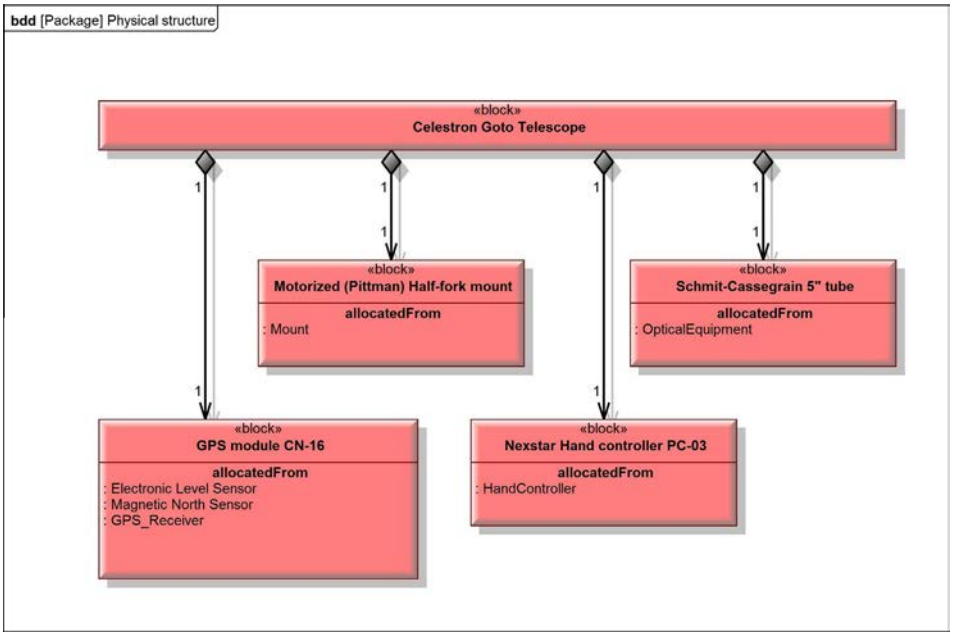


Fig. 18. The physical allocation (physical structure).

instance to check the consistency of a set of performance requirements (see step 8).

4.4.6. Step 15: The physical allocation (physical structure)

After having checked with simulation (Modelica, Simulink, HIL) that the logical architecture is relevant to fulfill the set of requirements, it is time to choose physical components (components off-the-shelf (COTS), machined or molded parts, e.g. instances with suppliers references) and allocate them to logical components. The physical breakdown of the telescope is presented in Fig. 18.

4.4.7. Step 16: Physical architecture

An internal block diagram (IBD) allows specifying the interactions (flows) between the physical components. The physical architecture may be not similar to the logical architecture which is the case for the telescope (Fig. 19). In this case, consistency shall be maintained by allocating the physical components and ports to

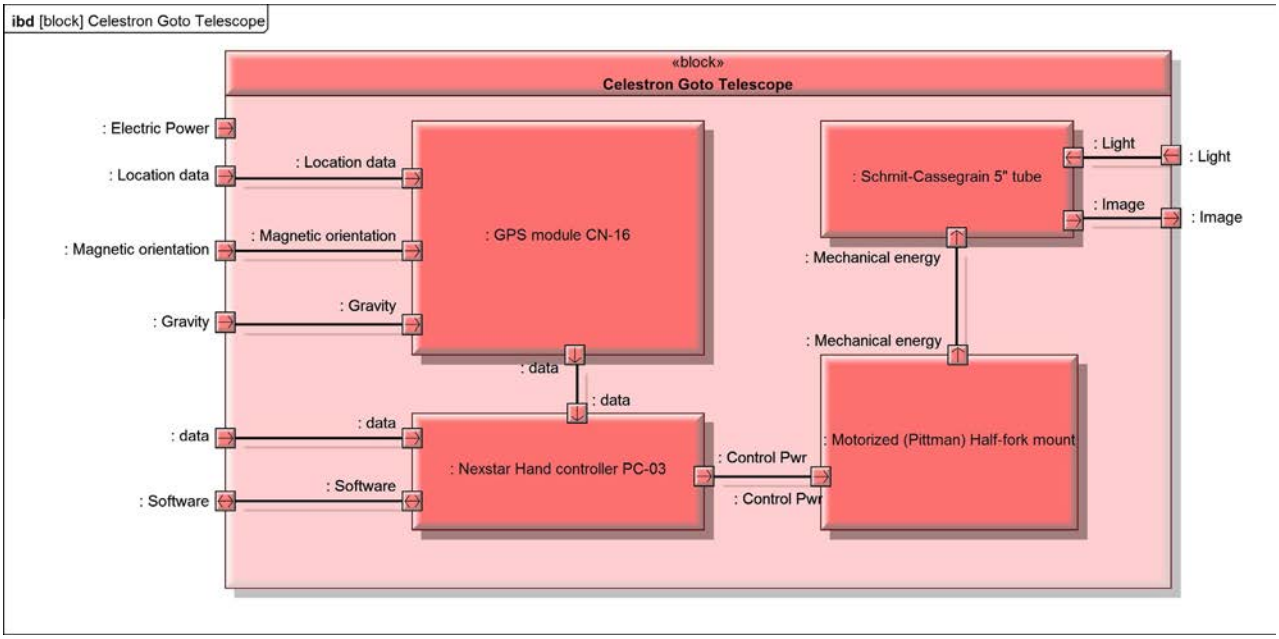


Fig. 19. The physical architecture.

the corresponding logical ones and justify if new components or ports are added.

5. Discussion

The proposed methodology gives a way to translate progressively user needs (or an initial set of requirements) into a feasible, well defined solution that takes into account the different aspects and constraints concerning the whole lifecycle of the system. A black-box analysis firstly aims at identifying a comprehensive set of requirements. A white-box analysis is then performed on order to define the system behavior and structure. Is should be bared in mind that it is question of an iterative process. Some diagrams may reveal information that was not considered in previous steps. the designer shall then go back to add the missing information and check all the impacts of this change on the model. The sequential aspect presented in this paper aims at making the understanding easier for the reader.

The presented methodology however, is not comprehensive and shall be extended with additional support processes. It is kind of the basic skeleton on which other engineering processes can be added. For instance, it does not yet offer processes for ranking and prioritizing requirements, establishing trade-offs and performing safety analyses.

Currently, the authors already work on these extensions. Developments about the integration of safety analysis into a SysML-based SE process are being done and some results are available in [17,18]. Other works on metrics are also being developed to help the designer to take decisions and trace his choices [29].

For a seamless process, some tools, like Model Center for instance, are developing bridges between SysML and other modeling and simulation tools. This is a good point that the authors are exploiting to further extend the current methodology.

6. Conclusion

In this paper, a SysML based methodology for mechatronic systems design is proposed. This method is made up of two main phases. A black-box analysis phase with the objective of collecting all the requirements and necessary data that will be a baseline for the next design phase. A white-box analysis phase is then performed to identify and compare candidate solutions and then select the final physical architecture of the system. The methodology is iterative and, at each stage, designers can go back and modify previous steps. The adequacy of SysML to hold these steps is demonstrated with a case study example.

Detailed design simulations and trade-off studies however are performed in parallel with this process with other dedicated tools. The main advantage of the methodology and of the use of SysML is a unique system model shared by all the contributors. Mechatronic systems may interact and communicate together in a kind of "internet of things" to form Cyber-Physical Systems (CPS). In our future work, we consider to extend the current methodology to support CPS modeling.

Acknowledgements

This work has been partially supported by the Linz Center of Mechatronics (LCM) in the framework of the Austrian COMET-K2 program.

References

[1] G. Ferretti, G. Magnani, P. Rocco, Virtual prototyping of mechatronic systems, Annu. Rev. Control 28 (2004) 193–206.
[2] P. Hehenberger, Advances in Model Based Mechatronic Design, Trauner Verlag, 2012.
[3] D. Shetty, R.A. Kolk, Mechatronics Systems Design, Global Engineering: Christopher M. Shortt Publisher, Cengage Learning Custom Publishing edition, 2011.
[4] P. Hehenberger, F. Poltschak, K. Zeman, W. Amrhein, Hierarchical design models in the mechatronic product development process of synchronous machines, Mechatronics 20 (2010) 864–875. Special Issue on Theories and Methodologies for Mechatronics Design.
[5] A.A. Cabrera, M. Foeken, O. Tekin, K. Woestenenk, M. Erden, B.D. Schutter, M. van Tooren, R. Babuka, F. van Houten, T. Tomiyama, Towards automation of control software: a review of challenges in mechatronic design, Mechatronics 20 (2010) 876–886. Special Issue on Theories and Methodologies for Mechatronics Design.
[6] INCOSE, INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, International Council of Systems Engineering, Cecilia Haskins edition, 2006.

[7] W.B. Rouse, Engineering complex systems: implications for research in systems engineering, IEEE Trans. Syst. Man Cybernet. Part C: Appl. Rev. 33 (2003) 154–156.
[8] I. 1220, Ieee standard for "application and management of the systems engineering process", 1999.
[9] E. 632, Processes for engineering a system, 1998.
[10] ISO/IEC-15288, Systems engineering – system life cycles processes, 2005.
[11] SYSML, OMG Systems Modeling Language (OMG SysML), 2012.
[12] OMG-SYSML, www.omgsysml.org, 2013.
[13] D. Chapon, G. Bouchez, On the link between architectural description models and modelica analyses models, in: Proceedings 7th Modelica Conference, Como, Italy, September 20–22, 2009.
[14] E. Adrianarison, J.-D. Piques, SysML for embedded automotive systems: a practical approach, in: Embedded Real Time Software and Systems ERTS 2010, 2010.
[15] R. Laleau, F. Semmak, A. Matoussi, D. Petit, A. Hammad, B. Tatibouet, A first attempt to combine SysML requirements diagrams and B, Innovat. Syst. Softw. Eng. 6 (2010) 47–54.
[16] P. David, V. Idasiak, F. Kratz, Reliability study of complex physical systems using SysML, Reliab. Eng. Syst. Saf. 95 (2010) 431–450.
[17] F. Mhenni, J.-Y. Choley, A. Rivière, N. Nguyen, H. Kadima, SysML and safety analysis for mechatronic systems, in: Mechatronics (MECATRONICS), 2012 9th France–Japan & 7th Europe–Asia Congress on and Research and Education in Mechatronics (REM), 2012 13th Int'l Workshop on Research and Education in Mechatronics, IEEE MECATRONICS REM, Paris, November 2012. ISBN: 978-1-4673-4772-3.
[18] F. Mhenni, N. Nguyen, H. Kadima, J.-Y. Choley, Safety analysis integration in a SysML-based complex system design, in: IEEE International Systems Conference "SysCon", Orlando, USA, April 2013.
[19] MOVEO, http://www.pole-moveo.org/pdf-projets-das/o2m-a.pdf, 2008–2010.
[20] J.A. Estefan, Survey of Model-Based Systems Engineering (MBSE) Methodologies, Technical Report, INCOSE MBSE Focus Group, 2008.
[21] H.-P. Hoffmann, Systems engineering best practices with the Rational Workbench for Systems and Software Engineering, IBM Corporation, Deskbook Release 3.1.1 edition, 2010.
[22] S. Friedenthal, A. Moore, R. Steiner, A practical Guide to SysML, The Systems Modeling Language, Morgan Kaufmann Publishers, 2009.
[23] IDEF0, Standard for Integration Definition for Function Modeling (IDEF0), 1993.
[24] J.E. Long, Relationships between common graphical representations used in system engineering, in: Proceedings of the SETE 2000, Systems Engineering and Test & Evaluation in the Changing Environment of the 21st Century, 2000.
[25] N. Belloir, J.-M. Bruel, N. Hoang, C. Pham, Utilisation de SysML pour la modélisation des réseaux de capteurs sans fil, in: Conférence sur les Langages et Modèles à Objets (LMO), Montréal, Canada, 03/03/2008-07/03/2008, Cépadués Editions, 2008, pp. 171–186.
[26] J.-D. Piques, E. Adrianarison, SysML for embedded automotive systems: lessons learned, in: Embedded real time Software and Systems ERTS.
[27] V. 2206, Design methodology for mechatronic systems, 2004.
[28] D. Firesmith, Common requirements problems, their negative consequences, and the industry best practices to help solve them, J. Object Technol. 6 (2007).
[29] A. Warniez, O. Penas, T. Soriano, About metrics for integrated mechatronic system design, in: Mechatronics (MECATRONICS), 2012 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM), 2012 13th Int'l Workshop on Research and Education in Mechatronics, IEEE MECATRONICS REM, Paris, November 2012. ISBN: 978-1-4673-4772-3.