How to define interface patterns in model-based system engineering

Indraja Elžbieta Germanaitė
Department of Information Systems
Kaunas University of Technology
Kaunas, Lithuania
e-mail: indraja.germanaite@ktu.lt

Prof. dr. Rimantas Butleris
Centre of Information Systems Design Technology
Kaunas University of Technology
Kaunas, Lithuania
e-mail: rimantas.butleris@ktu.lt

***Abstract*–The Patterns help system engineers to create efficient and more consistent System Models faster. Thus the methodology of quickly defining and describing the Patterns are needed. The review of the existing methodology of defining Interface Patterns is done in this article. As a result, the need for the new or more detailed ideas concerning the methodologies and implementations of the Patterns use could be identified.**

Keywords–Model-based System Engineering; MBSE Pattern; MBSE Framework; FAF; Viewpoint; SysML

I. INTRODUCTION

The discipline and practice of Model-based System Engineering (MBSE) follows the main concepts of the Systems Engineering discipline. The Systems Engineering covers a wide range of fundamental concepts such as system thinking, system science, life cycle management, system of systems, and agile and iterative methods [1]. System thinking captures and exploits what is common in a set of problems and corresponding solutions in the forms of various types. Thus, a pattern is a representation of similarities in a set of problems, solutions, or systems [1].

As already known from the object-oriented programming practice, the design patterns facilitate the reusing of successful designs and architectures, expressing proven techniques as patterns and making them more accessible to developers of new systems [2]. Patterns help to choose design alternatives that make a system reusable, to avoid alternatives that compromise reusability, and to improve the documentation and maintenance of the existing systems [2]. The patterns help the designer (or the system engineer) get the design (or the architecture) "right" faster.

The term "pattern" appears repeatedly in the history of design, such as civil architecture, software design, and systems engineering [3]. In the MBSE context, a pattern could mean the whole framework that covers entire systems and is used as a re-usable, configurable model of whole domains or platform systems - whether formal platform management is already recognized or not - or only as smaller-scale element design patterns within them [3]. The aim of this article is to look into the methodology of defining and describing subsystem or component patterns. In this case, the review of Pattern-Based Systems Engineering methodology based on S*Models and S*Patterns and dedicated to the complex systems is not in the scope of this article.

II. WHAT IS THE PATTERN IN MBSE?

John Holt et al. give the full picture of what is the MBSE Pattern and describe the main terms that are used for defining and describing the Patterns in their book [4]. The Model abstracts the System and is made of one or more Views that are the fundamental building blocks of the Model. The View uses a Systems Modeling Language (SysML) diagram as a representation, and is made of the View Elements that in turn use SysML elements as a representation. The View is the realization of a Viewpoint and the end product of the Pattern. The MBSE Framework provides the basis for the structure of the Model and defines a number of Viewpoints. The Viewpoint is a template for the Pattern that forms the basis for the View. In turn, the Viewpoint element is a template for the View Element. The Ontology is made up of one or more Ontology Elements that define the content of the Viewpoint. The main entities that participate in Pattern definition and relationships between those entities are shown in Figure 1.

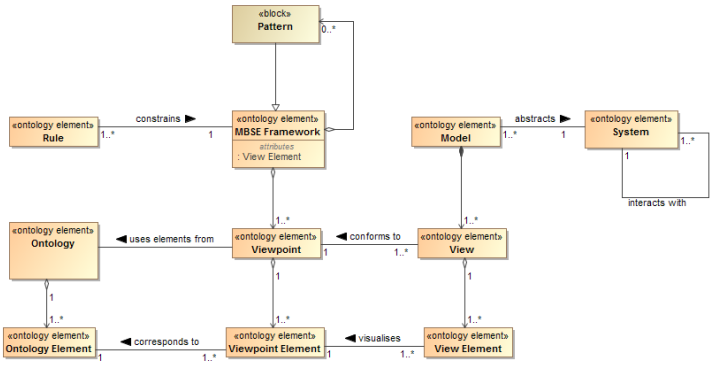

Fig. 1. The relationship between the System, the Model, The Framework, the Ontology, and the Pattern (taken and modified from [4]).

John Holt et al. [4] explain that a Pattern is a special type of the Framework and is made up of one or more Viewpoints, each of which comprises one or more Viewpoint Elements. The Viewpoint and Viewpoint elements are based on the Ontology, providing consistency with the rest of the Model. The main difference between a Framework and a Pattern is that a Framework has a single purpose and single application, while a pattern has a single purpose but multiple applications. For example, the Interface Pattern may be used as part of several Frameworks, at different level of abstractions, and in different

aspects of the Model [4]. Thus, the Pattern uses a template itself – the Viewpoint - and also serves as a template for the View. In addition, the Pattern could be understood as a Framework, but the Framework may also use many different Patterns for different purposes. So, the intent is to create an initial 'ontology' of high-level Pattern descriptions that provide a framework within which Pattern relationships can be discussed [5].

III. Defining Patterns

Going deeper into the definition of a Pattern, John Holt et al. [4] declare that the approach used to define each Pattern is the same as the approach used to define the Framework, and uses Framework for Architecture Frameworks (FAF). The FAF was defined to address the key questions through the MBSE approach based around the ideas of Ontology, Viewpoints, and Framework, and consists of the Ontology, six Viewpoints, and supporting processes [4]. The six Viewpoints directly address one of the six key questions that should be considered in order to approach the Pattern definition in a consistent and repeatable way [4]. These key questions with the attached FAF Viewpoints and Views are shown in Table I. It should also be noted that the Views may be visualized using any notation, text, tables, or formal techniques, and in any way that is appropriate [4]. The SysML is selected as the MBSE Pattern-defining language that is important and useful for its graphical notation.

TABLE I. FAF KEY QUESTIONS (TAKEN AND MODIFIED FROM [4])

Key question	FAF Viewpoint used to answer	View	Is derived from	SysML diagram used
What is the purpose of the Pattern?	Architectural Framework Context Viewpoint	Architectural Framework Context View		Use Case Diagram
What concepts must the Pattern support?	Ontology Definition Viewpoint	Ontology Definition View	Architectural Framework Context View	Block Definition Diagram
What different ways of considering the identified concepts are required to fully understand those concepts?	Viewpoint Relationships Viewpoint	Viewpoint Relationships View	Ontology Definition View	Block Definition Diagram
What is the purpose of each Viewpoint?	Viewpoint Context Viewpoint	Viewpoint Context View	Architectural Framework Context View	Use Case Diagram
What is the definition of each Viewpoint in terms of the identified concepts?	Viewpoint Definition Viewpoint	Viewpoint Definition View		Block Definition Diagram
What rules constrain the use of the Pattern?	Rules Definition Viewpoint	Rules Definition View		Block Definition Diagram

IV. The "Interface Definition Pattern" example

John Holt et al. give many examples of MBSE Patterns defined according to the FAF method in their book [4]. One of the most common examples is the "Interface Definition Pattern" chosen to illustrate MBSE Pattern definition in this article because it can be used in both software and hardware systems. More than that, interfaces form an integral part of any systems model and define a contract between system elements, whether those elements are physical or are realized in software [4]. Thus, the key aim of the Interfaces Pattern is the identification of interfaces and their relation to the system elements that use them and the ports that expose them [4].

According to the FAF method for describing the "Interface Definition Pattern", the main aims of the "Interface Definition Pattern" should be defined first. These aims are shown in the Architectural Framework Context View in the form of SysML Use Case Diagram in Figure 2.

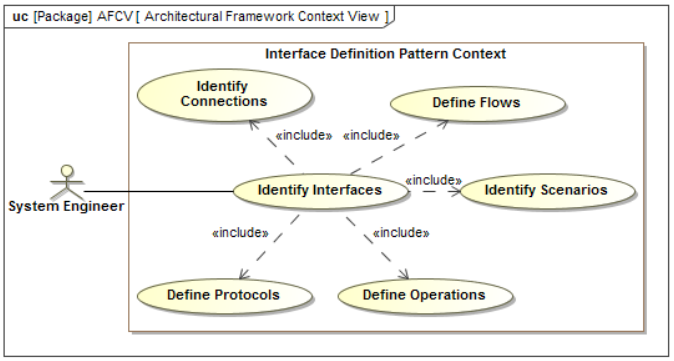

Fig. 2. Architectural Framework Context View (taken from [4]).

Second, the main concepts covered by the "Interface Definition Pattern" should be defined in the Ontology Definition View in the form of the SysML Block Definition Diagram. John Holt et al. define the main Interface Pattern concepts in their book [4]: an Interface has a Direction, which may take the values "in", "out", and "inout". The Direction property shows the direction in which the Interface operates from the point of view of the Port, owned by a System Element, which exposes the Interface [4]. Each Interface is described by an Interface Definition that defines the operations of a Service-Based Interface and the items transferred by a Flow-Based Interface [4]. A Port represents the interaction points between one or more System Elements and may represent the concept of a software Port or a physical Port, such as the connector for the fuel line on a car engine fuel pump [4]. Ports are connected to each other via a Port Connection. Ports and Interfaces may conform to one or more Protocols that describe and control how the Port and the Interface behave [4].

Third, the Viewpoints that make up the "Interface Definition Pattern" should be described. This should be done in the Viewpoint Relationships View in a form of the SysML Block Definition Diagram. Types of Pattern relationships are introduced as a mechanism to categorize and standardize basic relationship types and range from informal to formal [5]. The "Interface Definition Pattern" provides three Viewpoints that enable the identification and definition of Interfaces to be specified in terms of the structural aspects of the Interfaces: the

Interface Identification Viewpoint identifies each Interface, the Interface Connectivity Viewpoint shows the connection between Interfaces, and the Interface Definition Viewpoint defines what is transferred across each Interface [4]. The Pattern also provides two Viewpoints that enable the behavior of Interfaces to be specified: the Interface Behavior Viewpoint identifies typical scenarios showing how Interfaces are used, and the Protocol Definition Viewpoint defines any Protocols to which Interfaces or Ports must conform [4]. These Viewpoints will be described in more detail later in this article.

Fourth, the Rules that apply to the "Interface Definition Pattern" should be defined in the Rules Definition View in a form of the SysML Block Definition Diagram. The example of such rule could be the statement: "*Any protocol-based Interface or Port must have a corresponding Protocol Definition View defined*" [4].

V. The Viewpoints of the "Interface Definition Pattern"

John Holt et al. in their book provide a detailed description of the Viewpoints that make up the "Interface Definition Pattern" [4]. The "Interface Definition Pattern" consists of five Viewpoints that represent the structural or behavioral aspects of the Interfaces. Each of these Viewpoints consists of the Viewpoint Context View that defines the purpose of the each Viewpoint and a Viewpoint Definition View that covers the definition of each Viewpoint. Only the Viewpoint Definition View diagrams are presented later in this article, and Viewpoint Context Views will be discussed in the text as they describe only the aims of each Viewpoint.

The first Interface Identification Viewpoint identifies the System Elements, the Ports that they own, and the Ports that expose the Interfaces and the Interfaces that they expose [4]. By using the Interface Identification Viewpoint, it is possible to define both Service-Based and Flow-Based Interfaces. The examples of the Service-Based and Flow-Based Interface Identification Views are shown in Figure 3 and Figure 4, respectively.

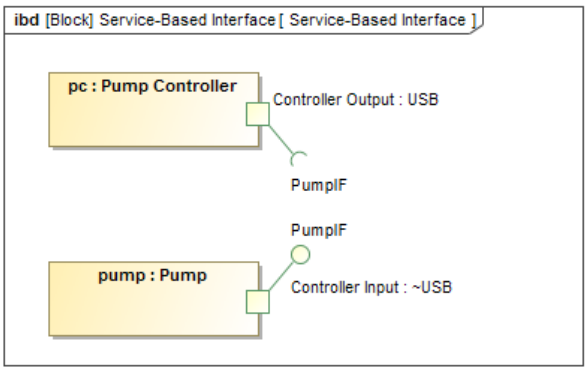

Fig. 3. Service-Based Interface Identification View (taken from [4])

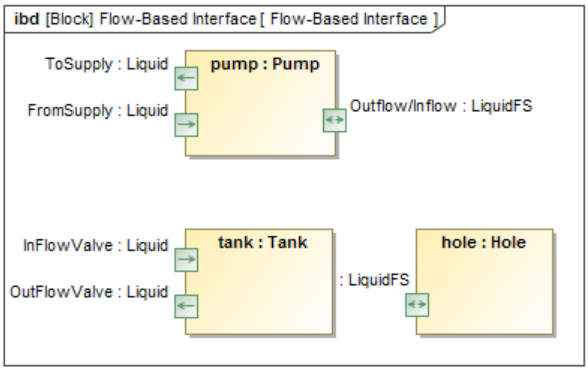

Fig. 4. Flow-Based Interface Identification View (taken from [4])

The second Interface Connectivity Viewpoint identifies connections between Ports and the Interface Connections that take place across the Port Connections. Again, two examples of Service-Based and Flow-Based Interface Connectivity Views are shown in Figure 5 and Figure 6, respectively.

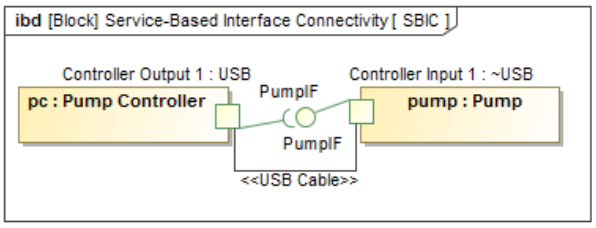

Fig. 5. Service-Based Interface Connectivity View (taken from [4])

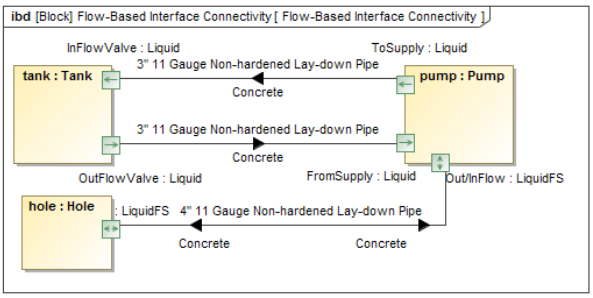

Fig. 6. Flow-Based Interface Connectivity View (taken from [4])

The third Interface Definition Viewpoint defines the operations or flows of data, material, energy, personnel, etc. of the Interfaces. An example shown in Figure 7 demonstrates the Service-Based Interface Definition (PumpIF), and the Flow-Based Interface Definition (LiquidFS) in one View.

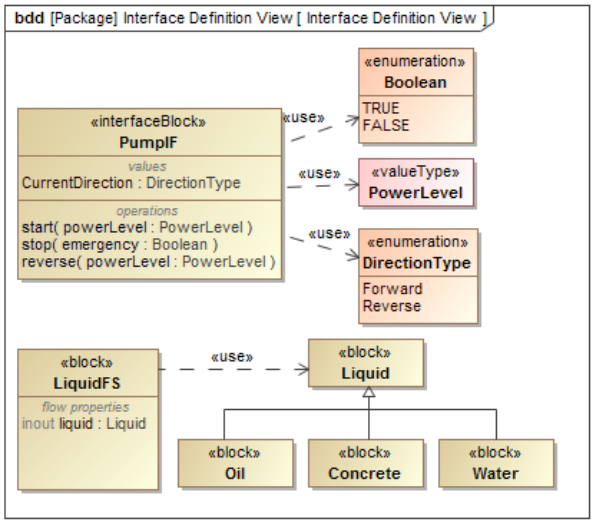

Fig. 7. Service-Based and Flow-Based Interface Definition View (taken from [4])

The fourth Interface Behavior Viewpoint identifies the typical scenarios showing how Interfaces are used [4]. An example of the Interface Behavior View is shown in Figure 8.

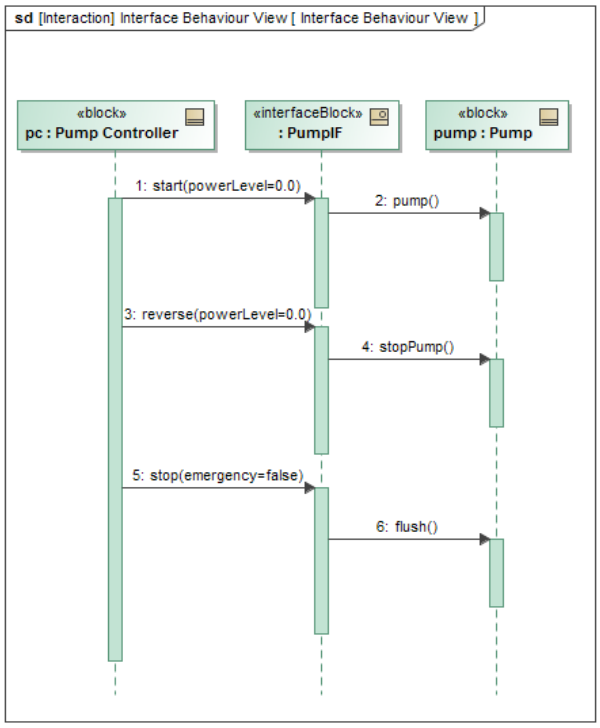

Fig. 8. Interface Behavior View (taken and modified from [4])

The fifth Interface Protocol Definition Viewpoint defines any protocols to which an interface or port must conform [4]. The Interface Protocol Definition View is represented by the SysML State Machine Diagram.

When using the "Interface Definition Pattern", at least one Interface Context View and one Interface Definition View are needed to specify Interfaces, their associated Ports, and the connections between them [4]. In practice, however, multiple Interface Identification Views, Interface Context Views, and Interface Definition Views would be produced along with the Interface Behavior View and Protocol Definition Views [4]. The discussed Viewpoints show that the context and the scope of the Viewpoint that is used for the View generation should be set: every Viewpoint must provide a template for answering some small set of core questions about the System for the customer stakeholders, which could not be otherwise easily answered [6]. The Viewpoint must identify and meet the needs of the target audience and their specific concerns, and must be well scoped and make the generated Views easy to draw inferences from [6].

VI. The Use of the MBSE Patterns

The main goal of the use of Patterns, as stated by John Holt et al. [4], is to allow re-use, which brings a number of tangible benefits. Among such benefits, John Holt et al. distinguish the shortened development time, since by working to a set of pre-defined Viewpoints, the structure and the content of each View has already been decided, hence reducing the amount of time required to generate the individual Views [4]. In addition, the improved consistency of the Model is emphasized because when working in accordance with the Pattern Viewpoints (and Ontology), the consistency between the resultant Pattern View is guaranteed [4]. If systems modelers are educated and trained in the use of Patterns, they can apply this knowledge in a variety of applications, thus shortening the learning curve [4]. John Holt et al. [4] also mentioned the benefit of increased efficiency through the use of the MBSE modeling tool for the implementation of Patterns, as the Viewpoints that describe the Pattern may be implemented in a modeling tool through the use of profiles.

When discussing the realization of the MBSE Patterns, John Holt et at. [4] stated that three aspects need to be considered in order to realize MBSE Patterns effectively and efficiently: People, Process, and Tools. For example, the MBSE Patterns can be used as part of the Competency assessment Process or to shorten the learning curve associated with MBSE [4]. The MBSE Frameworks may be constructed by re-using and tailoring the established Patterns, and if each Pattern has its associated Viewpoints defined, then the definition of the Framework becomes so much quicker and easier as the structure and contents of each View will already be defined [4]. Good MBSE modeling tools may be tailored to support a defined MBSE approach by defining the profiles, and thus the use of profiles allows the tool to be tailored to implement a specific approach to MBSE [4].

VII. Conclusions

The review of the existing methodology for the definition and description of the MBSE Pattern based on the book by John Holt et al. [4] showed that:

1. A consistent MBSE Pattern can be defined with the help of the Ontology and the Framework, using Viewpoints and Views as their realization.

2. In order to define the Pattern in a completed way, the key questions about the context, purpose, definition, relationships,

and rules of the Pattern should be answered in the form of SysML diagrams.

3. The structural and behavioral aspects of the Patterns are presented in five Viewpoints, which serve as a basis for the creation of Views.

4. The use of Patterns not only helps to shorten system development time, but also improves the consistency of the model, shortens the learning curve, and increases the efficiency of MBSE modeling tools.

5. The MBSE methodology should be used together with the MBSE Patterns that increases the effectiveness and efficiency of modeling.

REFERENCES

[1] Incose, INCOSE Systems Engineering Handbook, WILEY 2015.

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J., "Design Patterns." Addison-Wesley Publishing Company, 1996.

[3] Peterson, T., Schindel, W. D., Hamilton, B. A., "Model-based system patterns for automated ground vehicle platforms," INCOSE international symposium, vol. 25, pp. 388–403, 2015.

[4] Holt, J., Perry, S., Brownsword, M., "Foundations of model-based systems engineering: From patterns to models, Institution of Engineering and Technology." Iet Professional Applications of Computing, 2016.

[5] Simpson J., J., Simpson M., "Foundational Systems Engineering (SE) Patterns for a SE Pattern Language," INCOSE International Symposium, vol. 16, pp. 1675-1687, 2006.

[6] Paredis, C. J. J., Bishop, C., Bodner, D., "Introduction to Information Visualization (InfoVis) techniques for Model-Based Systems Engineering," Conference on Systems Engineering Research, vol. 16, pp. 49-58, 2013.