Sebastian Eder
Jakob Mund
Andreas Vogelsang

6

Logical Viewpoint

This section provides an outline of the logical viewpoint. This viewpoint describes the internal logical structure and the behavior of the system under development (SUD). The main task in the logical viewpoint is the distribution of functions to a hierarchy of logical components. The main model type of the logical viewpoint is the logical component architecture: it describes the logical components of the system and their relationship. The structure of this logical component architecture is often influenced by nonfunctional criteria, e.g., maintainability or reliability. In contrast to the technical viewpoint, the logical viewpoint does not focus on the technical infrastructure provided, e.g., the controllers or communication devices used.

K. Pohl et al. (eds.), *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*,
DOI 10.1007/978-3-642-34614-9_6,

6.1 Introduction

The logical viewpoint describes the logical structure and the distribution of responsibilities functionality of a system by means of a network of interacting logical components that are responsible for a set of functions. These logical components and their interactions are arranged in the logical component architecture of the system. The design of the logical component architecture is driven by various considerations, such as achieving maximum reuse of already existent components or fulfilling different nonfunctional properties of the SUD. The logical component architecture bridges the gap between functional requirements and the technical implementation. All examples in this chapter are taken from a case study modeled in SPES 2020 [Eder et al. 2011].

6.2 Concerns

Aims of the logical viewpoint

The main aims of the logical viewpoint are:

- Describing the internal logical structure of the SUD by partitioning the system into communicating logical components
- Allocating desired functions to cohesive logical units
- Supporting the reuse of already existent logical components and designing the logical components such that future reuse is facilitated
- Defining the total behavior of the system (as opposed to the partial behavior specifications in the models of the functional viewpoint) and enabling the complete simulation of the entire system

The logical components should be designed to capture the central domain abstractions and to support reuse. As a consequence, the logical component architecture should be as insensitive as possible to changes in the desired user functionality or technical platform. It should be the artifact in the development process with the highest stability and with the highest potential for reuse.

The functional black box model and the logical component architecture are two orthogonal structures of the system functionality. A brief comparison of both models is illustrated in Tab. 6-1.

Comparison of functional and logical viewpoints

In contrast to the functional black box model, the logical component architecture does not emphasize the formalization of the functionality that can be observed at the system boundary, but rather the structuring and partitioning of the SUD into communicating logical components. The behavior of these logical components as a whole realizes the

behavior determined by the functional viewpoint. The connection between these two models is established by the functional white box model of the functional viewpoint. A logical component represents a unit that provides one or more functions of the functional white box model. In other words: the functions of the functional white box model are distributed to logical components.

In the logical viewpoint, structuring is performed according to diverse criteria, such as the organizational structure within the company, nonfunctional requirements, or even according to the user function hierarchy of the functional black box model. However, it is important to note that the logical viewpoint abstracts from hardware details. Therefore, some (nonfunctional) requirements are better addressed in the technical viewpoint. Hint 6-1 summarizes the partitioning in the logical viewpoint.

Partitioning in the logical viewpoint

Hint 6-1*: Partitioning in the logical viewpoint*

- A core activity in the logical viewpoint is the partitioning of functions into communicating logical components
- Partitioning can be performed according to a user function hierarchy of the functional viewpoint, the organizational structure, or nonfunctional requirements
- Hardware details are part of the technical viewpoint rather than the logical viewpoint

The logical viewpoint provides a complete description of the system functionality, without, however, anticipating technical decisions with regard to implementation (e.g., the platform on which the logical components will be deployed).

Tab. 6-1 *Brief comparison of the functional black box model and the logical component architecture*

Functional Black Box Model	Logical Component Architecture
Problem domain	Solution domain
Black-box view of the SUD	White-box view of the SUD
Structured by user functions	Structured by architectural entities
Used primarily to specify what the SUD should do	Used primarily to design the SUD
Functional specification may overlap and must be checked for inconsistencies (horizontal decomposition)	Network of communicating logical components (vertical decomposition). Their composition must be checked against the desired system functionality.
Captures the functionality of the SUD	Works as a first cut at design
(Possibly) partial behavioral specification	Total behavioral specification

6.3 Logical Component Architecture

Logical components

The logical component architecture is the main model of the logical viewpoint and consists of logical components. These logical components are decomposed into other logical components, which results in an acyclic hierarchical structure of logical components, i.e., a tree. The leaves of this tree are called atomic logical components and they are not decomposed further.

Similarly to the models of the functional viewpoint, every logical component carries a syntactic interface consisting of typed ports that are either used for output or input. Input ports can be connected to output ports by channels and thus, logical components are linked together via channels.

Ports and channels

Logical components define these channels to connect their logical subcomponents. Every atomic logical component has to specify its (total) behavior. The behavior of an atomic logical component is defined directly (for example, using an automaton). The behavior of non-atomic components is derived from the composition of their subcomponents.

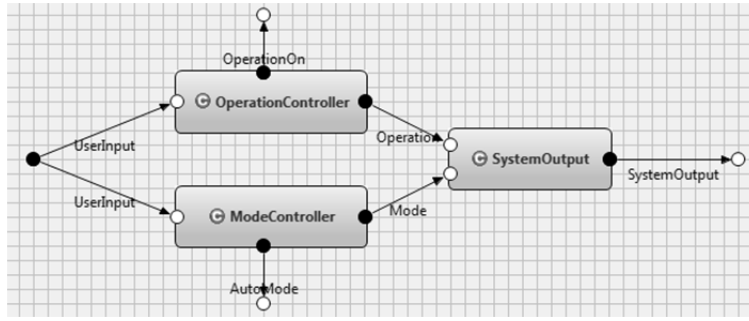

Fig. 6-1 *Example of a logical component architecture*

Example 6-1: User input controller

Fig. 6-1 shows an example of a logical component architecture that is part of the model of the logical viewpoint. The architecture models a user input controller consisting of three logical components: OperationController, ModeController, and SystemOutput. The output ports are illustrated as black circles placed at the borders of the logical components. White circles depict input ports. The ports not drawn at the border of any logical component represent input or output ports to the next higher logical component in the hierarchy. Output ports are connected to input ports via channels, as depicted by the arrows. The labels next to the channels associate a name with the corresponding channel. Note that each channel has a specific type, but this is not shown in Fig. 6-1.

Hint 6-2*: Using the metamodel of the logical viewpoint*

- Entities for the distribution of functions are logical components that are the basic building blocks representing the behavior.
- Each logical component offers a syntactic interface that consists of (typed) ports. Ports can furthermore be input or output ports for a given logical component.
- Communication between logical components is achieved by connecting output ports to input ports by means of channels for which the types of the ports must coincide.

Usage guidelines for the logical viewpoint

6.4 Analyses

The logical viewpoint provides a model of the system design independent of any hardware decision. However, this model is expressive enough to allow a variety of analyses to be performed on it. Ultimately, the logical viewpoint aims at constructing models that are so close to the final implementation that the code can be generated completely from the models of the logical viewpoint. Further analyses that can be performed on the models of the logical viewpoint are:

- *Complete simulation:* Since the model of the logical viewpoint defines a total system behavior, it allows comprehensive simulation of the SUD. While a simulation of the functionality is already possible in the functional viewpoint, the logical viewpoint also allows several nonfunctional properties to be checked (e.g., reliability and abstract timing constraints) by simulating the SUD.
- *Verification of system properties:* Properties that have been specified previously (e.g., in the functional viewpoint) can be verified automatically using model checking techniques.
- *Test case generation:* As the model defines the total behavior of the SUD, this enables test cases to be generated, for example, by following [Pretschner et al. 2004].
- *Concurrency analyses:* For a couple of years, there has been an ongoing paradigm shift from single-core towards multicore processors. Employing multicore architectures for embedded systems brings several advantages but also poses new challenges for software engineering. The logical viewpoint allows analysis of the logical component architecture in order to determine an adequate parallel hardware platform and an adequate deployment of logical components to hardware components. Thereby, adequacy can refer to several system requirements (e.g., response time, robustness, or hardware costs).

6.5 Integration in the SPES Modeling Framework

Within the SPES modeling framework, the logical viewpoint resides between the functional and the technical viewpoints. This section gives an overview of the relation and the differences between the logical viewpoint and its adjacent viewpoints.

6.5.1 Functional Viewpoint

Relation to the functional viewpoint

The functional white box model of the functional viewpoint describes an abstract solution of the system. The goal is to realize the desired functionality as specified in the functional black box model using a set of abstract functions. The logical components of the logical component architecture comprise a set of these abstract functions in order to structure them. Therefore, the functions of the functional white box model must be distributed to the logical components of the logical viewpoint. In addition to this direct relation, it is also possible to relate the logical components to the user functions of the functional black box model by assessing the logical components that are involved in the realization of a user function. This relation can be used for impact analyses and other validation methods (cf. [Vogelsang et al. 2012]). Further details on the correlation between a user function and a logical component can also be found in Section 5.6.

6.5.2 Technical Viewpoint

Relation to the technical viewpoint

The technical viewpoint describes the hardware topology of the system using technical components such as control units and busses. Logical components defined in the model of the logical viewpoint are deployed on the technical components described by the model of the technical viewpoint, and communication channels between logical components are realized either internally on one controller or by busses or other technical communication facilities. Logical components are typically deployed as atomic units and are thus rarely split up into tasks that can then be seen as atomic units that are deployed. However, the modeling theory does not forbid this and this results in an n:m relation between logical components and technical components.

6.5.3 Abstraction Layers

Logical viewpoint and the abstraction layers

The relation of the logical viewpoint model over the abstraction layers can be described as follows: a change of the abstraction layer in the logical viewpoint corresponds to a decomposition of a logical component or the entire system into logical subcomponents. In this case, the logical viewpoint model serves as a structural characteristic for the lower abstraction layer, which means that, on the lower abstraction layer, viewpoint models exist for each logical subcomponent. The input and output data that is processed by the logical viewpoint model of a certain abstraction layer must also appear in the logical viewpoint model of the next lower abstraction layer; in fact they must be processed by the logical subcomponents. In more complex development scenarios, for example, when suppliers are involved, the change of an abstraction layer determined by one decomposition step of the system might not be handled that strictly. Instead, the transition to the next abstraction layer can, for example, be interpreted as handing over a certain part of the system to a supplier or another department within the company. In this scenario, several decomposition steps will be performed in one abstraction layer until the parts are handed over to a supplier or another department whose models reside in the next lower abstraction layer. However, the aforementioned relation between these abstraction layers also holds in this scenario. The supplier must provide viewpoint models for the specific part that is in his responsibility.

6.6 The Logical Viewpoint Process

Step 1: Adopt system boundary from the functional black box model

To develop the logical viewpoint of a system, the system boundary from the functional black box model is adopted and can be refined, if necessary. The system is the root logical component of the decomposition tree containing all logical components. The system boundary is expressed by a syntactic interface.

Step 2: Decomposition into logical subcomponents

The next step, which is one of the most crucial steps in developing the model of the logical viewpoint, is the decomposition of the SUD into logical components. The result of the decomposition is a tree of logical components, with the complete SUD as its root. This decomposition is oriented on nonfunctional requirements and quality aspects, and not solely on the functional requirements of the SUD. Thus, a logical component is not a user function, but a logical unit that is required to implement the system's functionality.

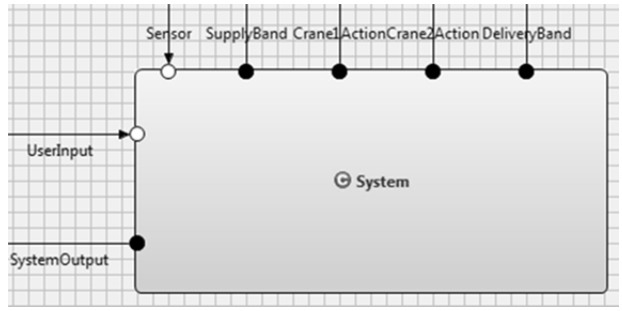

Fig. 6-2 *The system is the root logical component of the logical component hierarchy.*

Step 3: Connect components via channels

The different logical components of a system can be linked together via communication channels. These channels should only link logical components that really are intended to communicate. To enable logical components to be connected, every logical component is enriched by input and output ports that form the syntactic interface of the logical components. The ports are then used to connect logical components via channels.

> **Example 6-2**: Decomposition of the SUD
>
> Fig. 6-3 shows the decomposition of the SUD from Fig. 6-2. The system is decomposed into three logical components: IOAdapter, Transportation-Controller, and UserInteraction. The decomposition is guided by separation of concerns, separating the IO, the transportation unit and the user interface. The ports at the system boundary in Fig. 6-2 correspond to those ports in the decomposed network in Fig. 6-3 that do not belong to any of the logical components.

Step 4: Specify behavior for all leaf components

The logical components that are not decomposed any further (leaves in the logical component tree) are then enriched with behavior. This can be done using several techniques: table specifications [Thyssen and Hummel 2011], state machines [Broy and Stølen 2001], data flow diagrams [Leuxner et al. 2010], or mathematical functions, to name just a few. Unlike the models of the functional viewpoint, logical components define total behavior. This means that a logical component has to have behavior specified for every input. The composition of the behavior of all leaves results in the behavior of the complete system.

Therefore, the SUD can be simulated and the perceived behavior can then be compared to the specification of the system and the models of the functional viewpoint. If necessary, the logical component architecture can be refined or altered by running through the steps above.

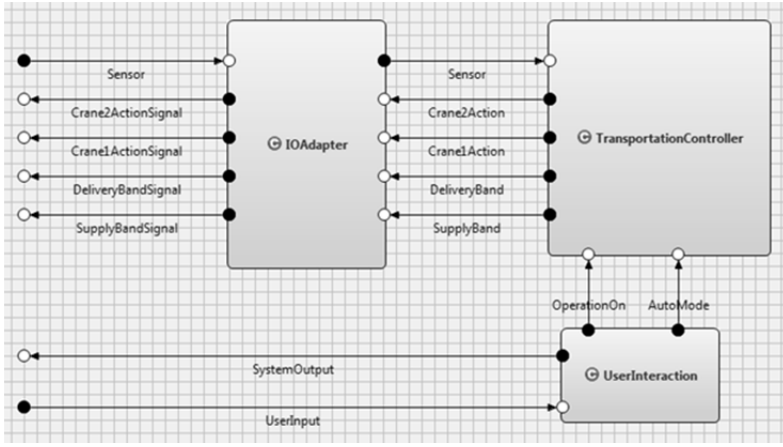

Fig. 6-3 *Decomposition of the SUD into a logical component architecture*

6.7 References

[Broy and Stølen 2001] M. Broy, K. Stølen: Specification and development of interactive systems: focus on streams, interfaces, and refinement. Springer, 2001.

[Eder et al. 2011] S. Eder, A. Vogelsang, M. Feilkas: Seamless modeling of an automation example using the SPES methodology. In: Technical Report TUM-I1110. Technische Universität München, May 2011.

[Leuxner et al. 2010] C. Leuxner, W. Sitou, B. Spanfelner: A formal model for work flows. In: SEFM 2010: Proceedings of the 8th International Conference on Software Engineering and Formal Methods, 2010.

[Pretschner et al. 2004] A. Pretschner, O. Slotosch, E. Aiglstorfer, S. Kriebel: Model-based testing for real. In: International Journal on Software Tools for Technology Transfer, Vol. 5, No. 2, 2004.

[Vogelsang et al. 2012] A.Vogelsang, S.Teuchert, J.-F. Girard. Extend and characteristics of dependencies between vehicle functions in automotive software systems. Proceedings of the 2012 International Workshop on Models in Software Engineering, 2012

[Thyssen and Hummel 2011] J. Thyssen, B. Hummel: Behavioral specification of reactive systems using stream-based i/o tables. Software and Systems Modeling, 2011.