Systems Engineering
Praktikum WS 23 – 24

Aufgabe 4 – Ganzheitliche Dokumentation und Konsistenzprüfung des Systementwurfs

Zielsetzung: Das Hauptziel dieser Aufgabe ist die Erstellung eines umfassenden, kohärenten und gut strukturierten Dokuments, das alle bisherigen Systementwürfe für das autonome Fahrzeug zusammenfasst. Sie sollen dabei besonderen Wert auf die Konsistenz und Integration der verschiedenen Entwürfe legen und diese ausführlich dokumentieren.

Aufgabenstellung:

1. **Konsistenzprüfung:**
    - Überprüfen Sie alle bisher erstellten SysML-Diagramme (Requirement Diagrams, Use Case Diagrams, Block Definition Diagrams, Aktivitätsdiagramme, Internal Block Diagrams, Parametrische Diagramme und Sequenzdiagramme) auf ihre Konsistenz.
    - Stellen Sie sicher, dass alle Entwürfe nahtlos ineinandergreifen und dass die in früheren Aufgaben definierten Anforderungen und Funktionen in den späteren Entwürfen adäquat berücksichtigt werden.
2. **Dokumentation und Beschreibung:**
    - Verfassen Sie für jedes Diagramm einen begleitenden Text, der die wichtigsten Elemente, deren Funktionen und die Beziehungen zwischen den verschiedenen Systemkomponenten erläutert.
    - Die Beschreibungen sollten präzise und detailliert sein, um ein klares Verständnis des jeweiligen Diagramms und seiner Rolle im Gesamtsystem zu ermöglichen.
3. **Verwendung von SysML Frame Headern:**
    - Nutzen Sie die SysML Frame Header, um eine klare Strukturierung und einfache Navigation innerhalb Ihres Gesamtdokuments zu gewährleisten.
    - Die Frame Header sollten so gestaltet sein, dass sie einen Überblick über den Inhalt des jeweiligen Diagramms bieten und dessen Einordnung in den Gesamtzusammenhang des Systementwurfs erleichtern.

4. **Integration und ganzheitliche Betrachtung:**
    - Stellen Sie eine Verbindung zwischen den einzelnen Diagrammen her, um ein ganzheitliches Bild des Systementwurfs zu schaffen.
    - Berücksichtigen Sie dabei, wie die verschiedenen Systemkomponenten und Prozesse miteinander interagieren und sich gegenseitig beeinflussen.
5. **Formatierung und Präsentation:**
    - Achten Sie auf eine klare, übersichtliche und professionelle Präsentation des Gesamtdokuments.

Abgabe:

- Ein digitales Gesamtdokument, das alle SysML-Diagramme mit begleitenden Texten und Frame Headern beinhaltet.

Nutzen Sie die angehängte Erklärung zu Frame Headern für diese Aufgabe

2.4 General Diagram Concepts

You should be aware of a few overarching concepts about SysML diagrams before you delve into the details of the specific types. A sample SysML diagram is shown in Figure 2.2.

Each diagram has a frame, a contents area (colloquially called the "canvas"), and a header. The diagram **frame** is the outer rectangle. The **contents area** is the region inside the frame where model elements and relationships can be displayed. The **header** is in the upper-left corner of the diagram, shown in Figure 2.2 with its lower-right corner cut off.

In SysML (unlike in UML), the frame must be displayed. With that said, some figures in this book show model elements and relationships without an enclosing frame. I do that to hide inconsequential information and focus your attention on particular notations. Officially, though, the frame is mandatory.

One of the most important diagram concepts is the format of the header information. The header commonly contains four pieces of information:

- Diagram kind
- Model element type
- Model element name
- Diagram name

The format of that information is shown in the header in Figure 2.3.

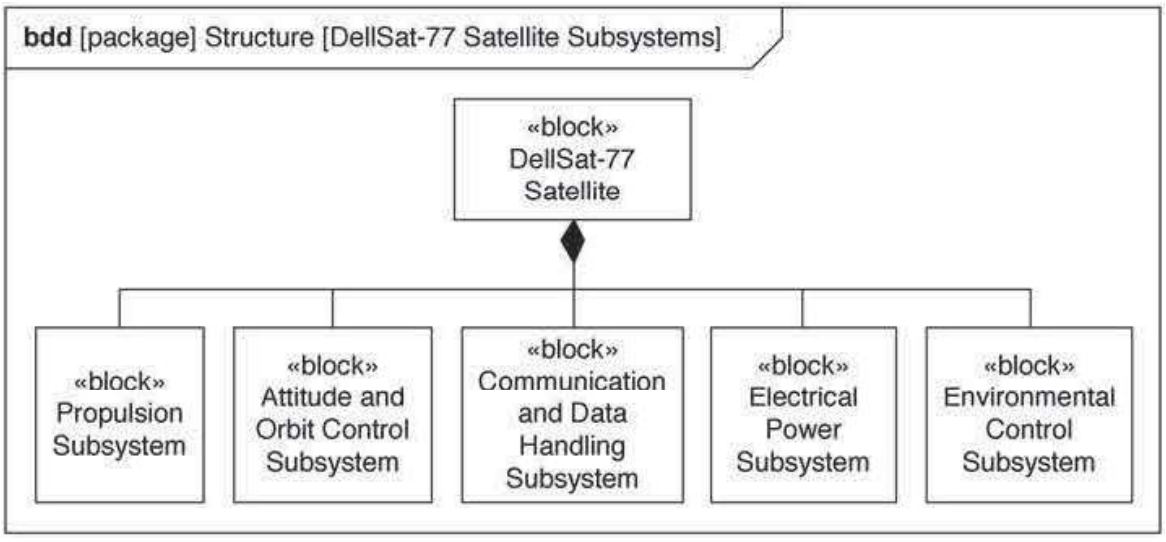

Figure 2.2 *Sample SysML diagram*

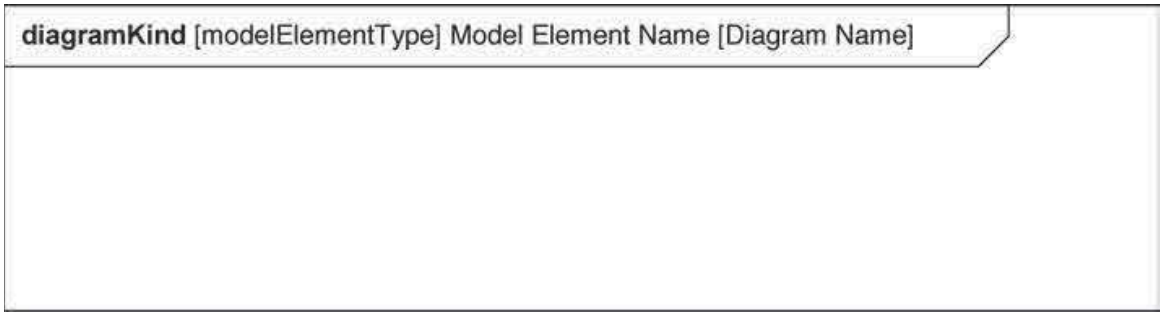

Figure 2.3 *Diagram header format*

I begin with two intuitive pieces of information: diagram kind and diagram name. The **diagram kind** is shown as its SysML-defined abbreviation:

- **bdd** = block definition diagram
- **ibd** = internal block diagram
- **uc** = use case diagram
- **act** = activity diagram
- **sd** = sequence diagram
- **stm** = state machine diagram
- **par** = parametric diagram
- **req** = requirements diagram
- **pkg** = package diagram

Based on this, you can conclude that the diagram in Figure 2.2 is a block definition diagram. (Chapter 3, "Block Definition Diagrams," discusses in detail the kinds of elements that can appear on a BDD.)

The **diagram name** can be anything you want it to be. I advise you to choose a diagram name that conveys which aspect of the model is in focus on that diagram. For example, the name of the diagram in Figure 2.2 is "DellSat-77 Satellite Subsystems." This diagram name indicates that the focus of the diagram is the set of subsystems that make up the satellite system. The model certainly contains other information about the satellite system, but that information is not the focus of this diagram.

The next two pieces of information in the header are the model element type and the model element name. To understand what these refer to, you first need to know another essential concept about SysML diagrams: Each diagram you create represents an element that you've

defined somewhere in your system model. More precisely, the diagram frame represents an element in the model. And *that* model element is the element whose type and name appear in the diagram header.

In Figure 2.2, the model element type is "package," and the model element name is "Structure." This conveys that the frame of this BDD represents the *Structure* package that exists somewhere in the system model hierarchy.

Requiring each diagram to represent a model element may seem like a strict and unnecessary constraint, but in the words of Frederick Brooks, Jr., in *The Design of Design* (Boston: Addison-Wesley, 2010), "Constraints are friends" (p. 127). The connection between a diagram and a model element was a deliberate and brilliant decision on the part of the SysML authors. The reason is conveyed by the next key concept of SysML diagrams: The model element represented by the diagram defines the **namespace**—the container element within the model hierarchy—for the other elements shown on the diagram. Simply put, the model element type and model element name shown in the diagram header indicate where the elements on the diagram can be found within the model.

The diagram header in Figure 2.2 tells us that the six blocks shown in the contents area are contained in (i.e., nested under) the *Structure* package in the model hierarchy. This gives us a sense of how elements are partitioned in the model and aids us in navigating it.

The model element may be a structural element (e.g., a package or block), or it may be a behavioral element (e.g., an activity, interaction, or state machine). The type of model element that a diagram can represent depends on the kind of diagram you're creating. The pairings are shown in Table 2.1.

Recall from Chapter 1 the distinction between a system model (as an engineering artifact in its own right) and the set of diagrams you create (which are views of the underlying model). This idea is so important that here I rephrase it more formally and give it a name: the fundamental precept of model-based engineering. Your assignment is to assume full lotus position and repeat the following mantra until you enter a deep meditative state:

> ***A diagram of the model is never the model itself; it is merely one view of the model.***

This idea is a paradigm shift for engineers who have only ever sketched designs using paper, whiteboards, or diagramming tools. And it's an idea best explained via metaphor: The model is a mountain,

Table 2.1 *Allowable Model Element Types for Each Diagram Kind*

Diagram Kind	Allowable Model Element Types
Block definition diagram	package, model, modelLibrary, view, block, constraintBlock
Internal block diagram	block
Use case diagram	package, model, modelLibrary, view
Activity diagram	activity
Sequence diagram	interaction
State machine diagram	stateMachine
Parametric diagram	block, constraintBlock
Requirement diagram	package, model, modelLibrary, view, requirement
Package diagram	package, model, modelLibrary, view, profile

and a diagram is a picture of the mountain. The mountain exists whether or not anyone takes a picture of it. If a man comes along and takes a picture from the north side, he creates one view of the mountain that shows some of its features but not others. If a woman takes a picture from the west side, she creates a second view of the mountain that shows a different—possibly overlapping—set of features.

Each of the two views focuses on a different aspect of the whole. But at all times the pictures are only views of the mountain and not the mountain itself. The mountain and its features will continue to exist even if the photographers later crop out certain details from the final pictures . . . and even if they destroy the pictures.

This metaphor fails in one respect: You can add new features and modify or delete existing features from the model at any time. When you modify an existing feature, the changes are instantly reflected on all diagrams that show the feature. When you delete a feature from the model, it instantly vanishes from all diagrams that showed that feature. Clearly the pictures in your photo album don't offer the same capability.

Earlier in this section I mentioned the practice of eliding inconsequential information on a given diagram. No diagram should attempt to convey every detail; the diagram would be unreadable. You should instead decide what you want the focus of a given diagram to be and then elide all model information that is not within that focus. This idea

leads to the corollary to the fundamental precept of model-based engineering:

> ***You cannot conclude that a feature doesn't exist from its absence on a diagram; it may be shown on another diagram of the model or on no diagram at all.***

Summary

SysML is a richly expressive graphical modeling language that you can use to visualize the structure, behavior, requirements, and parametrics of a system and communicate that information to others. SysML defines nine kinds of diagrams that you can use to convey all this system design information; each kind serves a specific purpose and conveys specific information about an aspect of a system.

The chapters that follow provide detailed coverage of these diagrams. You will learn the various kinds of SysML model elements—and the relationships among them—that can appear on each kind of diagram. I include discussions of the rules of SysML that you will need to know to build your system model correctly and ensure effective communication with your stakeholders.