Appendix C

Summary of diagramming guidelines

Concern for man and his fate must always form the chief interest of all technical endeavours. Never forget this in the midst of your diagrams and equations.

*Albert Einstein* (1879–1955)

C.1 Introduction

This appendix summarises the diagramming guidelines described in Chapter 6. No new information is added; indeed the descriptive text for each diagram is condensed in this appendix and the reader is directed to Chapter 6 for the full text.

C.2 Naming conventions

This section defines general naming guidelines that should be followed when producing SysML diagrams.

When modelling Standards etc. or producing models for customers, any naming conventions described in the Standard or used by the customer should be followed.

C.2.1 Structural diagrams

Figure C.1 illustrates the naming conventions to be followed when producing SysML structural diagrams.

The case of the text used in all elements indicates the convention to be adopted for that element. For example, an association should be named all in lower case whereas a property should be named in sentence case (i.e. initial word starts with a capital letter, all others with a lower case letter).

C.2.2 Behavioural diagrams

Figure C.2 illustrates the naming conventions to be followed when producing SysML behavioural diagrams.
Appendix omitted from Third Edition of “SysML for Systems Engineering” by Jon Holt and Simon Perry

708  SysML for systems engineering

The case of the text used in all elements indicates the convention to be adopted for that element. For example, a message should be named all in lower case whereas a use case should be named in sentence case (i.e. initial word starts with a capital letter, all others with a lower case letter).

![Diagram](image)

**Figure C.1** Naming conventions – structural diagrams
### C.2.3 Stereotypes

Figure C.3 illustrates the naming conventions to be followed when using stereotypes.

### C.3 Diagram frame labels

This section defines guidelines to be followed when labelling diagrams. All SysML diagrams must have a diagram frame that contains the name of the diagram. Each diagram should be named in the following fashion:

```
<frame tag> [model element type] <model element name> [diagram name]
```

Figure C.2 Naming conventions – behavioural diagrams
Each part is separated by a space and the frame tag is bolded. The model element type and diagram name parts of the name are in brackets. The frame tag and model element name are mandatory.

The abbreviations shown in Table C.1 should be used to indicate the type of diagram – known in SysML as the frame tag. If using a tool that automatically adds a diagram frame and that does not allow the frame tags to be changed, then the tag names used by the tool will be used.

The following shows the model element type associated with the different diagram kinds:

- activity diagram – activity
- block definition diagram – block, package, or constraint block
- internal block diagram – block or constraint block
- package diagram – package or model
- parametric diagram – block or constraint block
- requirement diagram – package or requirement

### Table C.1 Diagram frame labels

<table>
<thead>
<tr>
<th>Diagram type</th>
<th>Frame tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity diagram</td>
<td>act</td>
</tr>
<tr>
<td>Block definition diagram</td>
<td>bdd</td>
</tr>
<tr>
<td>Internal block diagram</td>
<td>ibd</td>
</tr>
<tr>
<td>Package diagram</td>
<td>pkg</td>
</tr>
<tr>
<td>Parametric diagram</td>
<td>par</td>
</tr>
<tr>
<td>Requirement diagram</td>
<td>req</td>
</tr>
<tr>
<td>State machine diagram</td>
<td>stn</td>
</tr>
<tr>
<td>Sequence diagram</td>
<td>sd</td>
</tr>
<tr>
<td>Use case diagram</td>
<td>uc</td>
</tr>
</tbody>
</table>
The *model element type* indicates the *namespace* for the elements contained on the diagram.

The *model element name* identifies which model element type the diagram is describing.

The *diagram name* is used to give the diagram a unique name. This is particularly important when different diagrams of the same type are drawn for the same model element. The diagram name would differentiate between these diagrams since they would have the same diagram kind, model element type and model element name.

For example, an Ontology View has been defined as part of an Architectural Framework meta-model. It has been given the abbreviation ONT and has the stereotype «ontology view» associated with it. The diagram is a usage of a SysML *block definition diagram*. An Ontology View is created in a *package* called MBSE Ontology and is intended to show a simplified Ontology. The *diagram frame* for this diagram would look like Figure C.4.

![Figure C.4 Example of diagram frame showing user-defined view abbreviation replacing frame tag](image)

In a tool that doesn’t allow replacement of *frame tags*, then the *diagram frame* would look like Figure C.5.

![Figure C.5 Example of diagram frame showing user-defined view abbreviation added to diagram name](image)
712  SysML for systems engineering

C.4  Additional guidelines

This section contains additional guidelines that apply to particular diagram types.

C.4.1  Block and internal block diagrams – showing interfaces

This section defines guidelines to be followed when producing block definition diagrams and internal block diagrams that show interfaces (Figure C.6).

```
Provided interfaces should be placed on the left of the block or part. If this is not possible, try to place them on the top of the block or part.

Required interfaces should be placed on the right of the block or part. If this is not possible, try to place them on the bottom of the block or part.

Where possible, place the interface label above the 'ball' or 'socket' symbol.

When a required and provided interface are connected on a diagram, use a single label.
```

Figure C.6  Block and internal block diagrams – showing interfaces

C.4.2  Block and internal block diagrams – showing item flows

This section defines guidelines to be followed when producing block definition diagrams and internal block diagrams that show item flows (Figure C.7).
Ports with incoming flows and ports with conjugated flows should be placed on the left of the block or part. If this is not possible, try to place them on the top of the block.

Ports with outgoing flows and ports with two-way, non-conjugated flows should be placed on the right of the block or part. If this is not possible, try to place them on the bottom of the block.

Port types can be omitted for single-direction ports only.

Figure C.7  Block and internal block diagrams – showing item flows
C.4.3 Activity diagrams

This section defines guidelines to be followed when producing activity diagrams (Figure C.8).

C.4.4 Use of colour

The use of colour can be used to add extra information to a diagram or to make diagrams clearer. However, colour should not be used without careful consideration. As a general rule, all diagram elements should be drawn with black text on a white background.

Figure C.8 Activity diagrams
Where colour is used, then the diagram must include a key that explains the
colour scheme and thought should be given to its choice and purpose. For example:

- How well will the colour reproduce when printed in black and white or
greyscale?
- Are any fonts or other symbols that lie on top of the colour readable?
- Will the chosen colours be problematic to those who are colour-blind?
- What meaning is attributed to the colours used?
- Have such meanings been made clear in a key?

In addition, some tools allow diagrams to be produced that have 3D effects,
gradient fills, shadows and curved lines that can be applied to diagram elements.
These should not be used, and should be turned off in the tool’s options.

C.4.5 Tool settings

Other common diagramming settings that need to be considered include the
display of navigability arrowheads and role names on associations, whether
whole-part relationships should default to composition or aggregation, whether
association names should be displayed by default, whether compositions, aggrega-
gations and generalisation relationships should be displayed in a tree layout, what
colours should be used for diagram elements such as blocks, requirements, and
use cases.

A SysML tool should allow such settings to be changed once for a model
and not force the modeller to change the settings for every diagram. Even more
desirable is the ability to define these settings for all models created with the tool.
This allows standard settings to be rolled-out across an entire organisation.
Unfortunately, not all tools allow changes to default settings to be made.

C.5 Model structure

When creating a SysML model it is important, in order to aid navigability and ease
of use, that the model is well structured. However, it is impossible here to define a
structure that is suitable for all projects; any structure adopted must be set up so as
to meet the needs of the project for which the model is being created. The authors
have created models that have been structured in many different ways. Some
examples of structuring adopted by the authors on projects include by:

- Life Cycle Stage
- Engineering process or activity
- System and subsystem
  - Structure
  - Behaviour
- Team
- Architecture framework
- Modelling framework
Sometimes model structure is a combination of these. For example, a model might first be structured by Life Cycle Stage, then within each Stage further structured by System. Figure C.9 shows part of a model of a Standard (ISO15288:2008 – see [ISO15288:2008] and the Process model in Appendix E) that is structured according to the seven views Process modelling Framework described in Chapter 7. Note the use of additional packages to contain aspects of the model such as styles (symbol colours etc.), stereotypes and scripts (the tool in which this model was produced allows the user to enhance functionality through user-defined scripts).

Another example is given in Figure C.10. Here, the model is structured largely into a structural and behavioural split influenced by engineering activity. For example, the ‘Coffin Escape Schematic’, ‘Requirements’, ‘Stakeholders’ and ‘Scenarios’ packages contain the parts of the model concerned with Requirements, whereas the ‘System’, ‘Constraints’, ‘Processes’ and ‘Units and Types’ packages contain the parts of the model concerned with design, defining System structure and behaviour.

Figure C.9 Example of model structured by modelling Framework Views

Figure C.10 Example of model structured largely by engineering activity and structural and behavioural split
Some SysML tools have a very useful facility that allows the model to be navigated both by the package structure (as in Figures C.9 and C.10) and by model Perspective and View. The two diagrams in Figures C.11 and C.12 illustrate this. These examples are taken from the Martian invasion case study model, discussed in Chapter 13.

The model structure shown in Figure C.11 is structured in a similar way to the model shown in Figure C.10, showing a structure based on a structural and behavioural split influenced by engineering activity.

As discussed in Chapter 13, the model has been constructed using an Architectural Framework that defines a number of Perspectives and Viewpoints (see Chapter 10 for a discussion of Viewpoints, Views and Perspectives). The model consists of a number of Views that conform to the Viewpoints (which are simply the definition of Views). Irrespective of the model structure defined by the user, the tool allows the model to be navigated by the Perspectives and Views defined by the Framework. An example is shown in Figure C.12, which shows the packages containing the Views that make up the System Perspective. The root package shows the Perspective and the sub-Packages correspond to each View in that Perspective (not all Views are shown). The View packages show all the model diagrams that conform to that View, no matter where in the package structure shown in Figure C.11 they reside. The Perspective and View structure is defined by and enforced by the tool; the user is not allowed to change this structure in any way.
Note that the package names for the View packages are lower case, which contradicts the naming convention described in section C.2.1. This is because the tool used requires the use of stereotypes to name Views in a way that makes them browsable as shown in Figure C.12; so the package labelled system structure view is displaying the all diagrams stereotyped «system structure view» and similarly for the others. The naming convention can thus be seen to be consistent with the guidelines for naming stereotypes given in section C.2.3.

Finally, many SysML tools will suggest a predefined model structure when a new model is created in the tool. While such structures may be of use in suggesting a starting point for the way the model is organised, they are rarely of much use beyond that. The model will be much easier to navigate if time is taken to define the structure that makes sense to the users of the model. The structure is up to you, but should be covered in your engineering Processes or modelling style guides.