ReqPat – Requirement Pattern Editor (RMF Version)

User Guide and Tutorial

FRAUNHOFER INSTITUTE FOR MECHATRONIC SYSTEMS DESIGN IEM

Zukunftsmeile 1
33102 Paderborn, Germany
Telephone: +49 5251 / 5465-101
Fax: +49 5251 / 5465-102
mechatronik@ipt.fraunhofer.de
www.ipt.fraunhofer.de/mechatronik

Created by
Markus Fockel and Jörg Holtmann

Paderborn, 7/15/2019

Contents

1 Introduction

ReqPat is an editor for requirement patterns. It is integrated within the Eclipse Requirements Management Framework (RMF) and the requirements editor ProR to assist in writing requirements using these patterns.

Section 2 shortly describes requirement patterns and the underlying specification methodology. Section 3 introduces the basic functionality of the tool and how to use it. Section 4 describes general information about the naming conventions to adhere to in ReqPat and a specification template with section headings and requirement patterns. Finally, Section 5 contains a short tutorial to guide an unexperienced user in the first steps of using ReqPat.

2 Requirement Patterns in a Nutshell

Requirement patterns are a means to textually describe the functionality of the system under development (SUD). Throughout this document, we use an automotive electronic control unit called *Body Control Module (BCM)* as a running example for the SUD. A BCM controls a multitude of vehicle body functions. We focus on the functionality of controlling the exterior lights as well as central locking.

2.1 Specification Methodology

The requirement patterns allow refining the overall system functionality by so-called *functions* across arbitrary abstraction layers, as depicted in Figure 1. The top-level function is refined into sub-functions to decompose the overall functionality of the system. Alongside, the input and output information of the top-level function is broken down to the sub-functions that process or create it. The functional decomposition results in a function hierarchy of the SUD.

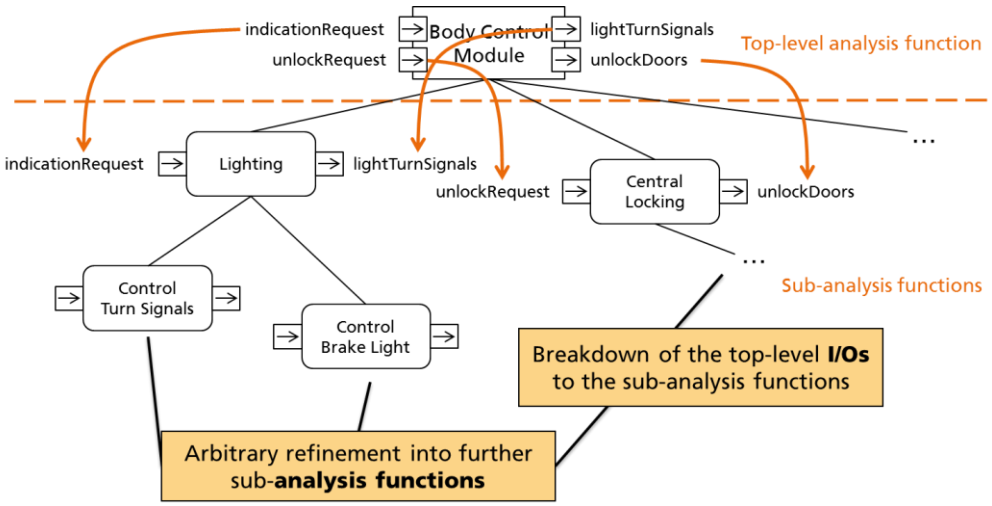

Figure 1: Visualization of the function hierarchy described by means of requirement patterns

2.2 Requirement Patterns

In the following all requirement patterns currently implemented in ReqPat are listed. The green parts in capital letters are variable parts and shall be replaced by appropriate element names. Parts in square brackets ('[' and ']') are optional parts of a pattern.

Function Hierarchy

The top-level function is called <NAME>.

The function <NAME> is a subfunction of the function <PARENTFUNCTION>.

Inputs and Outputs

The following information is received by the function <FUNCTION> [from the functional device <FUNCTIONALDEVICE>]: <INFORMATIONLIST>.

The following information is sent from the function <FUNCTION> [to the functional device <FUNCTIONALDEVICE>]: <INFORMATIONLIST>.

The following information is used by the function <FUNCTION>: <INFORMATIONLIST>.

The following information is created by the function <FUNCTION>: <INFORMATIONLIST>.

Purpose Description

The function <NAME> has the following purpose: "<PURPOSE>".

2.3 Grouping a Pattern-based Requirements Specification by means of Sections

Requirements are typically grouped by means of sections to enable structuring of and navigation within a requirements specification. ReqPat also applies this and uses the section hierarchy described in Section 4.2 to group requirements according to the functional decomposition. For the example depicted in Figure 1,

this hierarchy would look as follows:

1. Body Control Module
1.1 External Elements
1.2 Inputs & Outputs
1.3 Subfunctions
1.3.1 Lighting
1.3.1.1 Inputs & Outputs
1.3.1.2 Subfunctions
1.3.1.2.1 Control Turn Signals
1.3.1.2.1.1 Inputs & Outputs
1.3.1.2.2 Control Brake Light
1.3.1.2.2.1 Inputs & Outputs
1.3.2 Central Locking
1.3.2.1 Inputs & Outputs
1.3.2.2 Subfunctions
1.3.2.2.1 …
1.3.3 …

Each of the sections has a certain set of applicable requirement patterns as described in Section 4.2. Some examples are:

1. Body Control Module
`The top-level function is called BodyControlModule.`
…
1.2 Inputs & Outputs
`The following information is received by the function BodyControlModule: indicationRequest, unlockRequest.`
…
1.3 Subfunctions
1.3.1 Lighting
`The function Lighting is a subfunction of the function BodyControlModule.`
…

1.3.1.1 Inputs & Outputs
`The following information is used by the function Lighting: indicationRequest.`
…

The input/output information of the top-level function (e.g., Body Control Module) can be specified as being received from or sent to external systems (e.g., sensors,

actuators, other electronic control units). These external systems are called *functional devices* and would be described in Section "1.1 External Elements".

Based on the model of the function hierarchy that is described by the applied requirement patterns, additional properties can be specified by using further requirement patterns. Examples would be behavior descriptions or timing requirements.

3 Tool Usage

This section explains the usage of ReqPat.

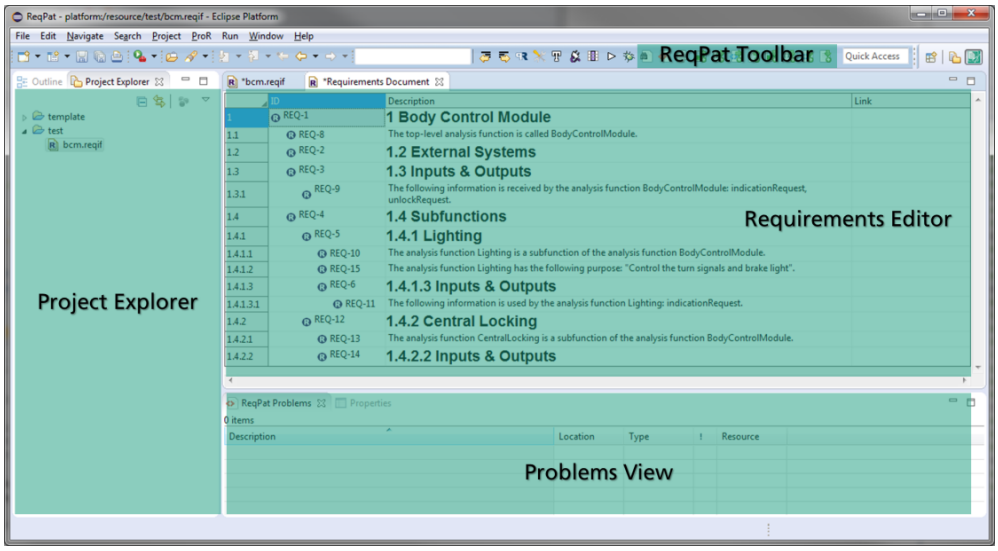

Figure 2 depicts an overview of the tool. You can find the project explorer as known from Eclipse to manage your files and projects on the left-hand side. The ReqPat toolbar, which is explained in Section 3.1, is located on the right side at the top. On the right-hand side in the middle, you can find the actual requirements editor. Section 3.2 explains how to edit the requirements using the editor. The problems view on the right-hand side at the bottom lists problems in the requirements specification found by ReqPat (cf. Section 3.3).

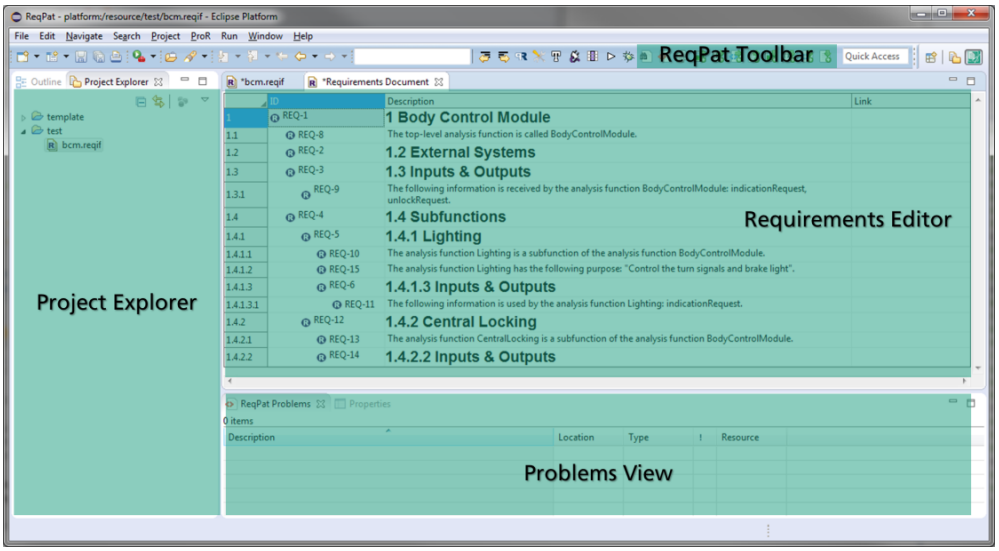

Figure 2: ReqPat overview

To create a new ReqPat requirements specification select the menu entry "File | New | Other..." and choose the "ReqPat Model" from the wizard (not the "Reqif10 Model") (cf. Figure 3).

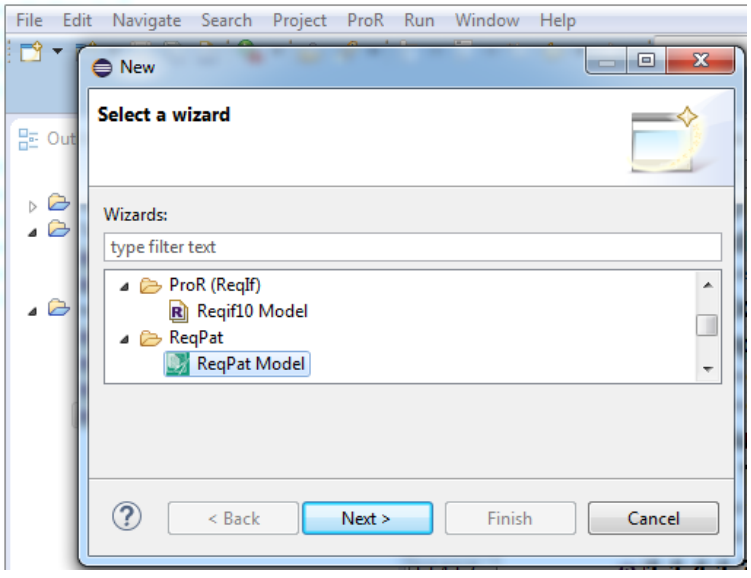

Figure 3: Creating a new ReqPat requirements specification

An overview of the requirements specification appears in the requirements editor as shown in Figure 4. Double click "Requirements Document" to start the actual editing of the requirements specification. Sometimes the "Requirements Document" already opens automatically.

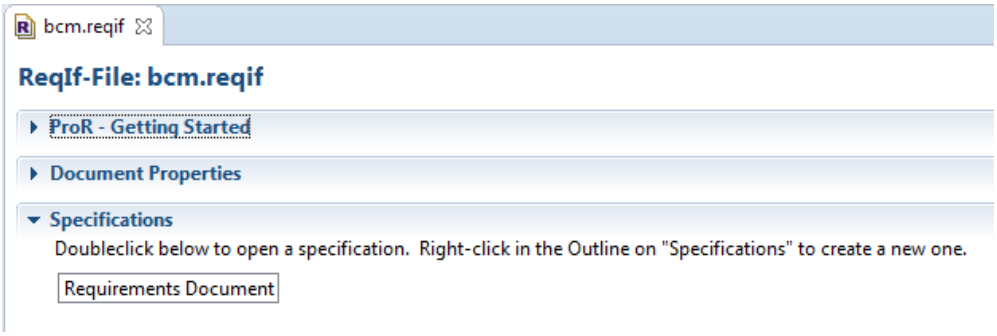

Figure 4: Requirements specification overview

3.1 ReqPat Toolbar

We explain the ReqPat-specific menu entries in this section. Note that there are several other menu entries that stem from Eclipse, RMF, and ProR that we do not explain here. Figure 5 depicts the ReqPat toolbar with numbered menu entries. In the following, we explain each of these menu entries according to its number.

Figure 5: ReqPat toolbar with numbered menu entries

1 **Find and Replace:** Allows to find and replace a string in the requirements specification.
2 **Check Document:** Checks the overall requirements specification for possible problems, see Section 3.3.
3 **Insert Template:** Adds a whole section from a template into the document below the currently selected entry, see Section 3.4.

4. **New Child (Image):** Inserts a new image below the selected entry in the requirements editor. The new entry will be one hierarchy level deeper than the selected entry.
5. **New Child (Heading):** Inserts a new heading below the selected entry in the requirements editor. The new entry will be one hierarchy level deeper than the selected entry.
6. **New Child (Requirement Type):** Inserts a new pattern-based requirement below the selected entry in the requirements editor. The new entry will be one hierarchy level deeper than the selected entry. See Section 3.2 how to edit pattern-based requirements.
7. **New Child (Free Text Requirement):** Inserts a new free text requirement below the selected entry in the requirements editor. The new entry will be one hierarchy level deeper than the selected entry. See Section 3.2 how to use free text requirements together with pattern-based requirements.
8. **New Sibling (Image):** Inserts a new image below the selected entry in the requirements editor. The new entry will be on the same hierarchy level as the selected entry.
9. **New Sibling (Heading):** Inserts a new heading below the selected entry in the requirements editor. The new entry will be on the same hierarchy level as the selected entry.
10. **New Sibling (Requirement Type):** Inserts a new pattern-based requirement below the selected entry in the requirements editor. The new entry will be on the same hierarchy level as the selected entry. See Section 3.2 how to edit pattern-based requirements.
11. **New Sibling (Free Text Requirement):** Inserts a new free text requirement below the selected entry in the requirements editor. The new entry will be on the same hierarchy level as the selected entry. See Section 3.2 how to use free text requirements together with pattern-based requirements.

Alternatively, you can use the context menu on a selected entry in the requirements editor to perform the actions 4-7 (see Figure 3.2) and 8-11 (see Figure 3.2) by doing a right click on the selected entry.

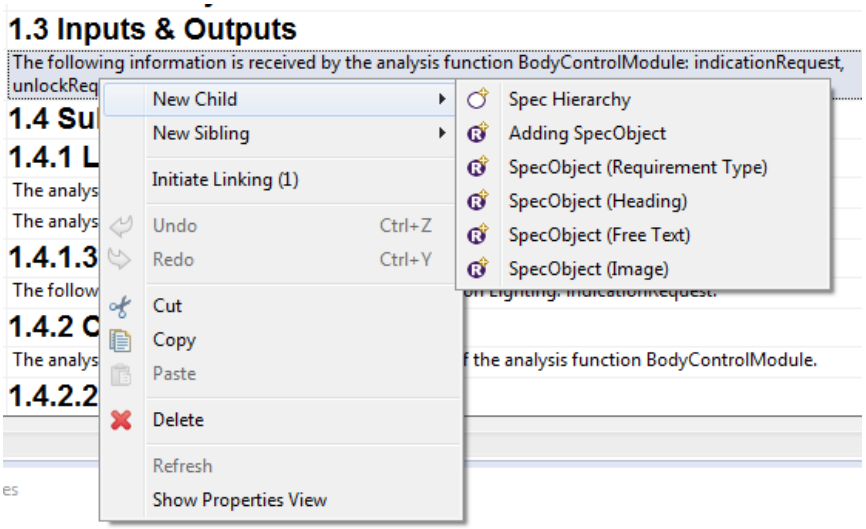

Figure 6: Context menu on a selected entry in the requirements editor – child-related actions

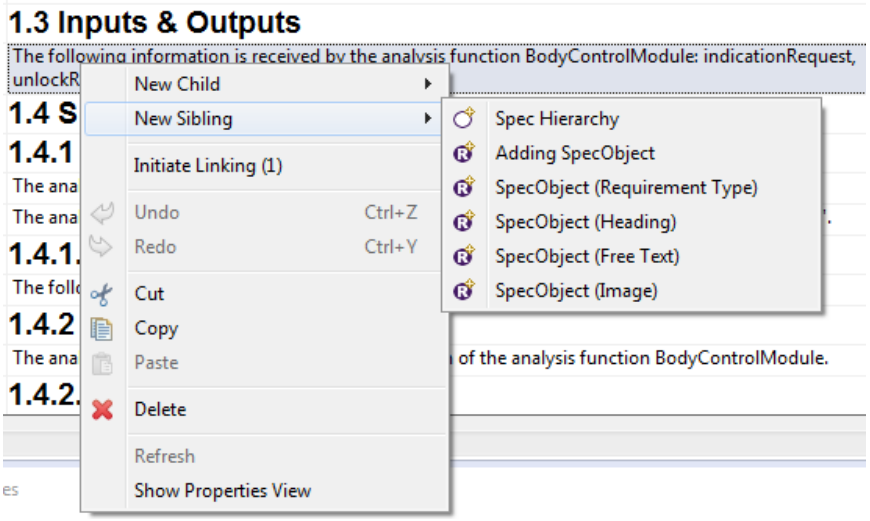

Figure 7: Context menu on a selected entry in the requirements editor – sibling-related actions

3.2 Edit Selected Requirement

For editing free text requirements, see Section 3.2.4. To edit a pattern-based requirement with the ReqPat editor, select it in the requirements editor and double-click it or use the shortcut "CTRL+SPACE". The ReqPat editor window will show up, as shown in Figure 8.

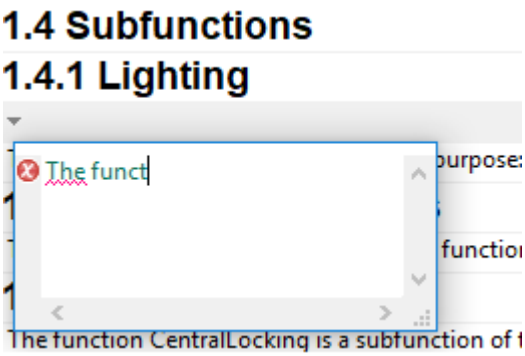

Figure 8: ReqPat editor window

For pattern-based requirements, ReqPat automatically checks for the correct application of requirement patterns directly while editing. Thus, problems are directly marked by a red wavy underline (cf. Figure 8). Note that sometimes intermediate states of the requirement can be marked as erroneous although the part of the requirement is correctly shaped w.r.t. a pattern, as it can be seen in Figure 8. This is due to the fact that the pattern is incomplete in this intermediate state.

For assistance with the current requirement pattern you can press "CTRL+SPACE" within the ReqPat editor window at any time. Afterwards, a list of proposals for different ways to work with requirement patterns will show up in a new window, as depicted in Figure 9. The entries 1 and 2 represent requirement patterns that are reasonable within the current section. The entries 3-6 propose individual parts of requirement patterns. The entries 7-16 represent all existing requirement patterns. The following three sections describe these different ways to edit pattern-based requirements, and Section 3.2.4 explains how to edit free text requirements.

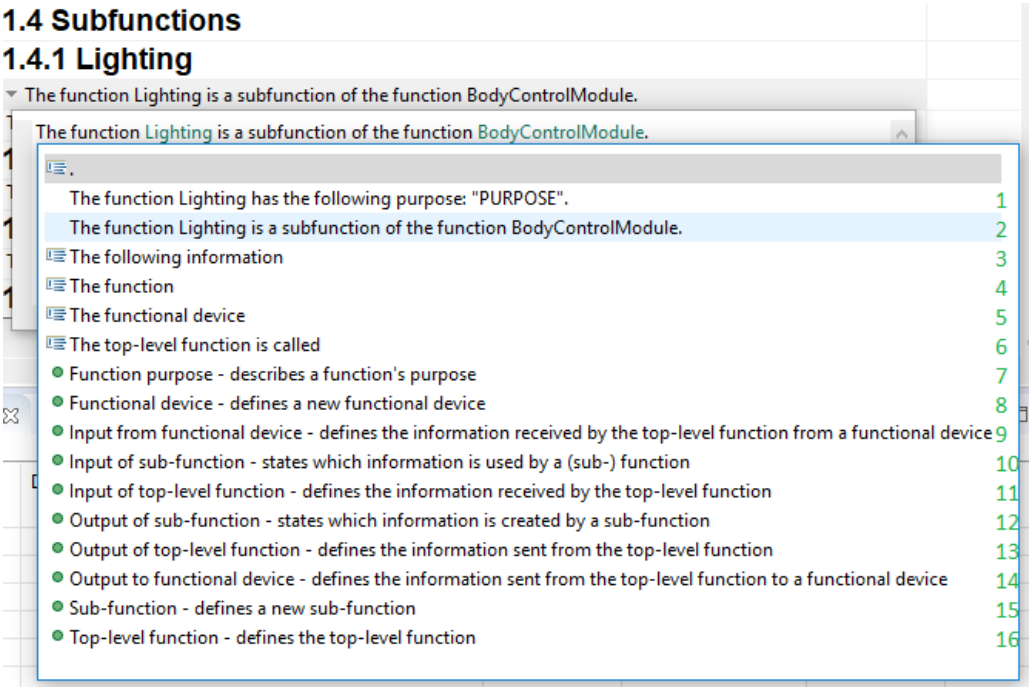

Figure 9: ReqPat content-assist

If you are at a variable part of a pattern-based requirement, you can call content-assist to fill the variable part with already defined functions or input/output information, as depicted in Figure 10. Additional information for each variable is displayed in a yellow window.

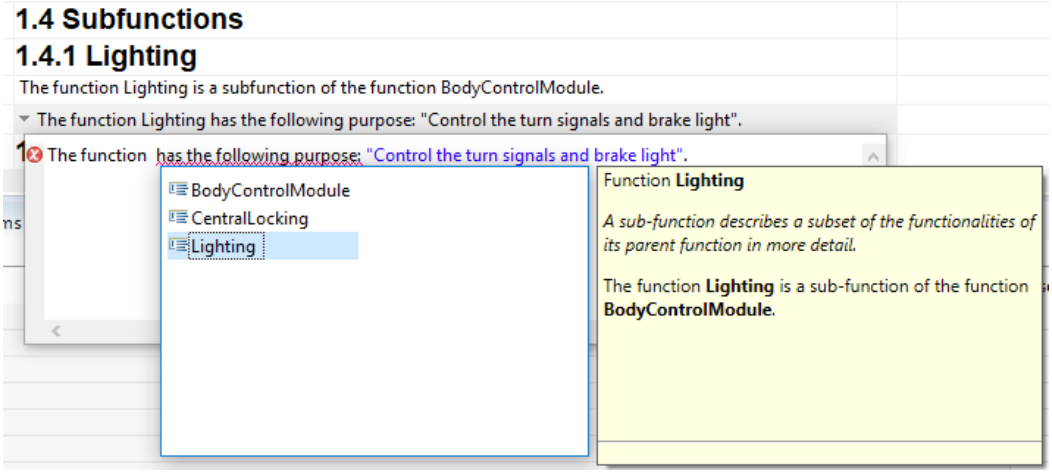

Figure 10: Content-assist for particular variables

Click somewhere outside the ReqPat editor window or press "CTRL+S" when finished. The ReqPat editor window disappears and updates the requirement in RMF automatically. If you want to discard your changes, press ESC.

When closing the ReqPat editor window by pressing "CTRL+S" the whole document will not be saved (the asterisk on the document's tab remains). This will require another press of "CTRL+S" when no ReqPat editor window is open.

3.2.1 Requirement Patterns Reasonable within Current Section

The first category of entries in the ReqPat content-assist window have no icon (see the first two entries in Figure 9) and represent requirement patterns that are reasonable within the current section. That is, only requirement patterns are proposed by ReqPat content-assist that are applicable within the current section based on the naming of the section's heading. Furthermore, this information is used to fill certain variable parts of the requirement pattern.

For example in Figure 9, since ReqPat knows that we are describing a subfunction "Lighting" of the overall functionality of "BodyControlModule" within the current section, the first two proposed patterns are filled with "The function Lighting…". Furthermore, the second entry is fully filled since it describes that "Lighting" is a subfunction of "BodyControlModule". This information can be fully derived by the section hierarchy.

See Section 4.2 for complete information about which patterns are reasonable in which specification sections.

3.2.2 Requirement Patterns Parts

The second category of entries in the ReqPat content-assist window has a grey icon representing text lines in front (see entries 3-6 in Figure 9). This category allows to write pattern-based requirements successively by means of individual parts of the patterns.

For example, if the third entry in Figure 9 is selected, the current requirement is filled with the corresponding pattern part. Afterwards, the name of the function "Lighting" has to be typed manually or already defined functions can be selected.

Now, two possible subsequent pattern parts can be chosen as shown in Figure 11. This procedure is repeated until the pattern-based requirement is complete.

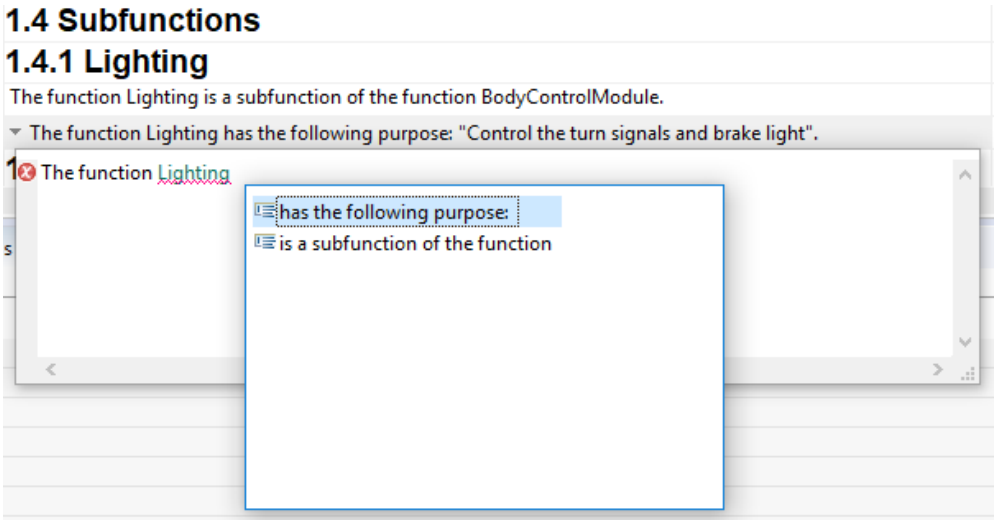

Figure 11: Content-assist for subsequent requirement pattern parts

3.2.3 All Requirement Patterns with Switching Between Variable Parts

Entries with a green dot in front represent the complete list (in contrast to patterns reasonable only in the current section) of whole patterns (cf. entries 7-16 in Figure 9). The particular entries do not display the actual patterns but the aspects they describe. For example, entry 7 states that you define the purpose of an function with the selected pattern.

After selecting one of these, you can switch between the variable parts of the sentence (in colored letters and framed by blue rectangles if selected) with the TAB key, as shown in Figure 12.

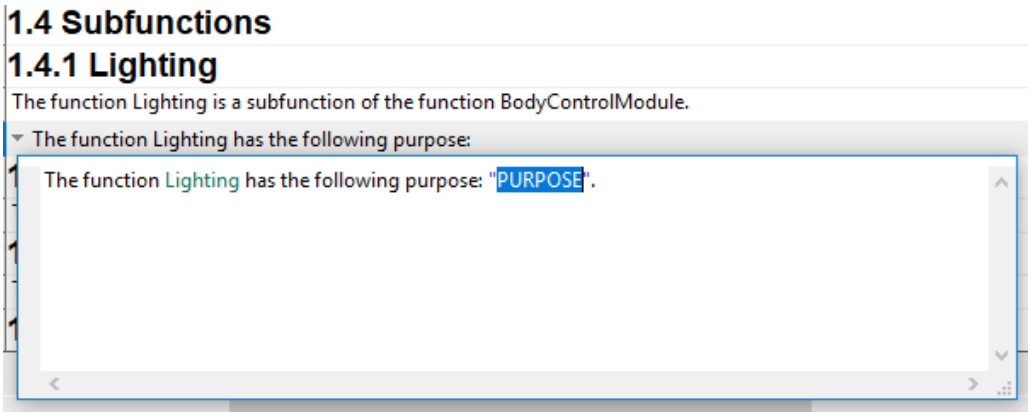

Figure 12: Switching between variable parts of a requirement pattern

3.2.4 Free Text Requirements

Editing free text requirements does not open an additional editor window but is done directly in the requirements editor, as depicted in Figure 13. Just hit RETURN to save or ESC to discard changes to the currently edited free text requirement.

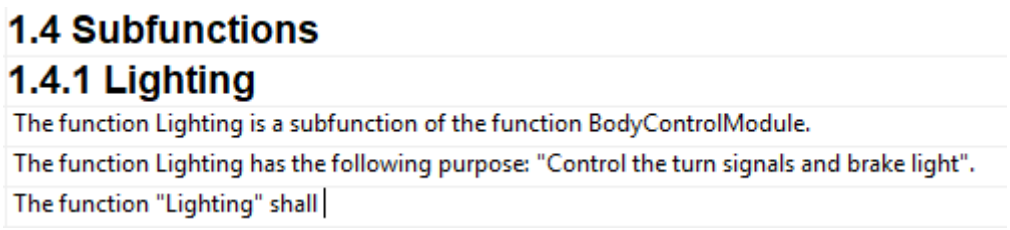

Figure 13: Editing free text requirements

You can explicitly refer to variables (i.e., functions or input/output information) defined by the pattern-based requirements by using single quotes (see "'Lighting'" in Figure 13). All texts that are within single quotes but do not refer to variables in pattern-based requirements are listed as problem if you check the document (cf. Section 3.3). This enables the consistent connection between pattern-based and free text requirements. Currently, ReqPat does not yet support content-assist for referring to variables out of free text requirements.

3.3 Check Requirements Specification for Possible Problems

Hit the icon 2 "Check Document" in the ReqPat toolbar (cf. Section 3.1) to check for possible problems in the requirements. A list of all problems with the current overall requirements specification shows up in the problems view as shown in Figure 14.

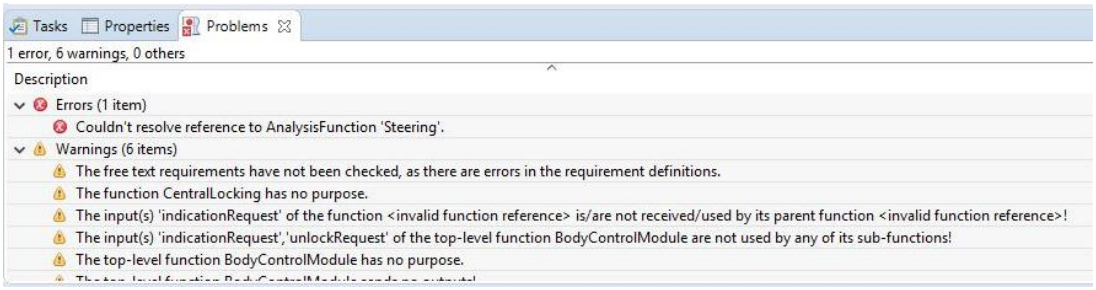

Figure 14: List of problems found by ReqPat in problems view

You can double-click a marked problem to solve it. Afterwards, the corresponding requirement entry will be selected in the requirements editor.

3.4 Insert Section from Template

Hit the icon 3 "Insert Template" in the ReqPat toolbar (cf. Figure 5 in Section 3.1) to add a whole section below the currently selected entry (typically a heading). The section to insert can be selected from a list of templates which are ReqPat Models (.reqif files) located in the "Templates" project (cf. Figure 15).

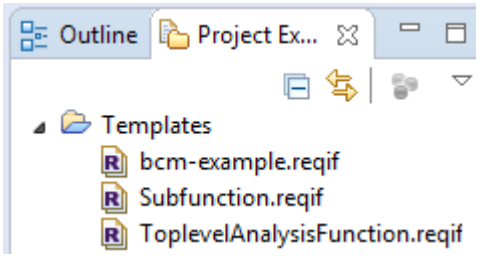

Figure 15: Section templates

bcm-example.reqif contains the example requirements specification used for the tutorial in Section 5. The other two files contain the sections for a top-level function and a subfunction, respectively, as defined by the specification template in Section 4.2.

To add a section for the subfunction "Lighting" of the top-level function "BodyControlModule" according to Figure 1, select the "Subfunctions" heading and hit the "Insert Template" icon 3 from the ReqPat toolbar. A wizard as seen in Figure 16 is showing up.

Via the "Select Template" button, the "Subfunction" template from Figure 15 can be selected. Afterwards, the "Key" column of the wizard's table is filled with placeholders that are defined in the template. These are words that can be replaced by text that is entered in the "Value" column. In Figure 17 the values for the "Lighting" subfunction from Figure 1 have been entered. After clicking the "Finish" button, the template's sections and requirements are added below the initially selected entry. The result is shown in Figure 18.

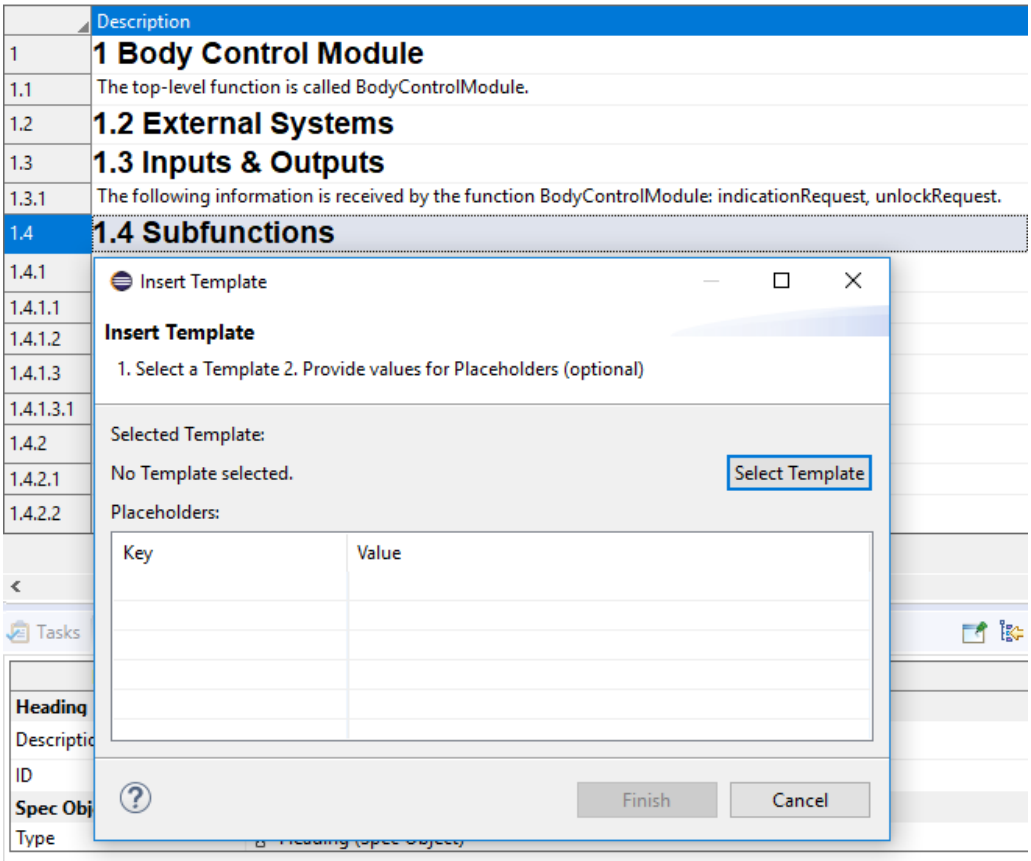

Figure 16: Insert Template wizard

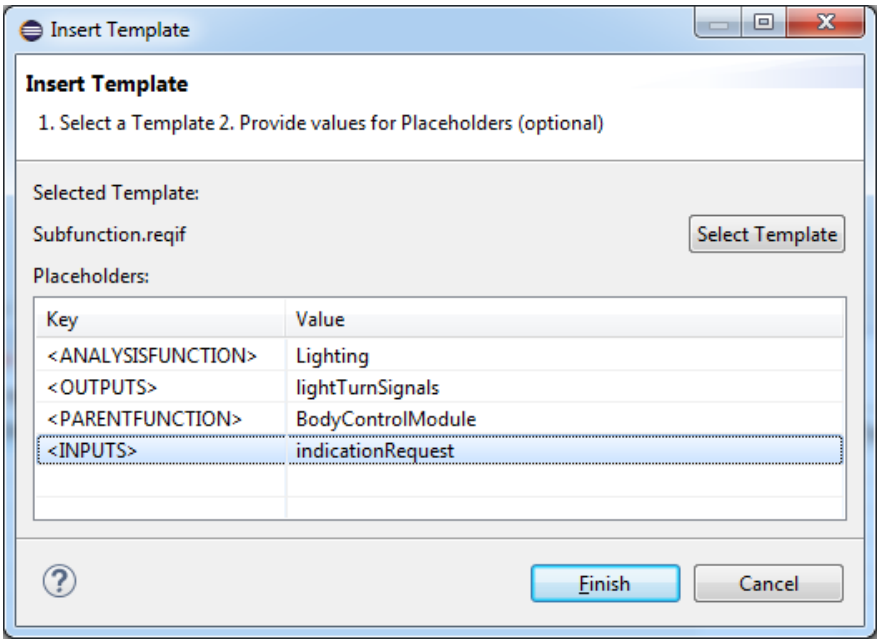

Figure 17: Insert template placeholders

	ID	Description
1	REQ-1	**1 Body Control Module**
1.1	REQ-8	The top-level function is called BodyControlModule.
1.2	REQ-2	**1.2 External Systems**
1.3	REQ-3	**1.3 Inputs & Outputs**
1.3.1	REQ-9	The following information is received by the function BodyControlModule: indicationRequest, unlockRequest.
1.3.2	REQ-17	The following information is sent from the function BodyControlModule: lightTurnSi
1.4	REQ-4	**1.4 Subfunctions**
1.4.1	REQ-5	**1.4.1 Lighting**
1.4.1.1	REQ-10	The function Lighting is a subfunction of the function BodyControlModule.
1.4.1.2	REQ-15	The function Lighting has the following purpose: "PURPOSE".
1.4.1.3	REQ-6	**1.4.1.3 Inputs & Outputs**
1.4.1.3.1	REQ-11	The following information is used by the function Lighting: indicationRequest.
1.4.1.3.2	REQ-16	The following information is created by the function Lighting: lightTurnSignals.
1.4.2	REQ-12	**1.4.2 Central Locking**
1.4.2.1	REQ-13	The function CentralLocking is a subfunction of the function BodyControlModule.

Figure 18: Result of inserting subfunction template

4 General Information

This section describes further general information about ReqPat. In the following section, naming conventions for the variable parts of requirement patterns are specified. The second section explains the specification template together with the requirement patterns applicable in each specification section.

4.1 Naming Conventions

Each element (function, information, …) is identified by its name. The name must begin with a letter and may only consist of letters, numbers, hyphens and underscores (no whitespaces or quotation marks).

4.2 Specification Template with Requirement Patterns

ReqPat's Content-Assist provides additional help based on the following template of section names and section hierarchy. It makes prefilled proposals for requirement patterns depending on where you are in the specification.

1 <Name of top-level function>
The top-level function is called <NAME>.

The function <NAME> has the following purpose: "<PURPOSE>".

1.1 External Elements
The functional device <NAME> has the following purpose: "<PURPOSE>".

1.2 Inputs & Outputs
The following information is received by the function <FUNCTION> from the functional device <FUNCTIONALDEVICE>: <INFORMATIONLIST>.

The following information is sent from the function <FUNCTION> to the functional device <FUNCTIONALDEVICE>: <INFORMATIONLIST>.
1.3 Subfunctions

1.3.1 <Name of sub-function>
The function <NAME> is a subfunction of the function

<PARENTFUNCTION>.

The function <NAME> has the following purpose: “<PURPOSE>”.

1.3.1.1 Inputs & Outputs

The following information is used by the function <FUNCTION>: <INFORMATIONLIST>.

The following information is created by the function <FUNCTION>: <INFORMATIONLIST>.

1.3.1.2 Subfunctions

1.3.1.2.1 <Name of sub-sub-function>

…

1.3.1.2.1…

1.3.1.2…

1.3.2 <Name of sub-function>

1.3.2…

1.3…

5 ReqPat – Tutorial

This short tutorial shall help an unexperienced user to use the features described in Section 3. The tutorial bases on a predefined requirements specification excerpt in the file "bcm-example.reqif" that was created according to the specification methodology explained in Section 2.1.

5.1 Preparations

1. Create a new project in the project explorer (File | New | Project… | General | Project).
2. Copy the file bcm-example.reqif from project "Templates" to the newly created project.
3. Open bcm-example.reqif in the newly created project.
4. Open the actual requirements specification by double-clicking "Requirements Document" (cf. Section 3). This step might happen automatically.

5.2 Edit the Requirements Specification

5. Select REQ-13 within section 1.4.2 "Central Locking".
6. Create a new pattern-based requirement on the same hierarchy level than REQ-13 via menu entry 10 of the ReqPat toolbar (cf. Figure 5 in Section 3.1) or via the corresponding context menu of the selected entry (cf. Figure 7 in Section 3.1).
7. Edit the newly created entry and open the ReqPat editor window via double-clicking the entry or pressing "CTRL+SPACE". You will see a window as in Figure 8 in Section 3.2.
8. Open the content-assist via "CTRL+SPACE". You will see a window as in Figure 9 in Section 3.2.
9. Select entry 7 from Figure 9 in Section 3.2 to describe the purpose of the function "CentralLocking".
10. The requirement pattern from Figure 12 in Section 3.2.3 is proposed. Jump with "TAB" between the variable parts of the pattern.
11. Jump to the variable part "NAME", call content-assist via "CTRL+SPACE", and select "CentralLocking".
12. Jump to the variable part "PURPOSE" and type "Control the door locks".
13. Save the requirement by clicking outside the ReqPat editor window or by pressing "CTRL+S".
14. Select REQ-14 in the bottom (section 1.4.2.3 "Inputs & Outputs")

15 Create a new pattern-based requirement one hierarchy level deeper than REQ-14 via menu entry 6 of the ReqPat toolbar (cf. Figure 5 in Section 3.1) or via the corresponding context menu of the selected entry (cf. Figure 7 in Section 3.1).
16 Edit the newly created entry and open the ReqPat editor window via double-clicking the entry or pressing "CTRL+SPACE". You will see a window as in Figure 8 in Section 3.2.
17 Type "The following" and call content-assist via "CTRL+SPACE". Only pattern parts and complete patterns are shown that fit to the already typed text (cf. Figure 19).

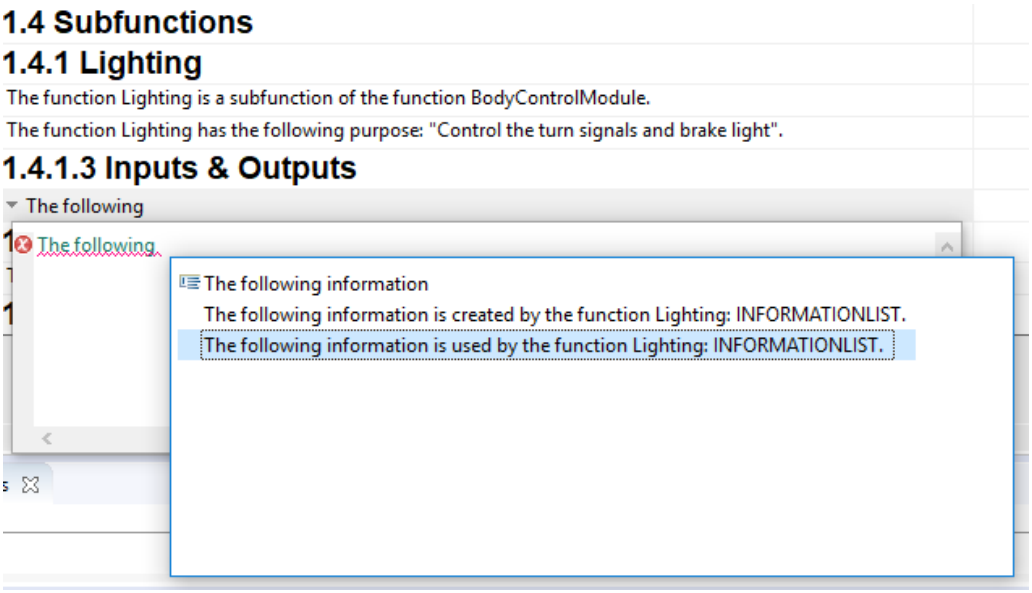

Figure 19: Content-assist based on already typed text

18 Select the last entry (cf. Figure 19).
19 Select "INFORMATIONLIST" by double-clicking on it.
20 Call content-assist via "CTRL+SPACE" and select "unlockRequest".
21 Type an arbitrary letter behind "unlockRequest", e.g., an "s" would lead to "unlockRequest**s**". The word is marked erroneous by a red wavy underline.
22 Despite the error, save the requirement by clicking outside the ReqPat editor window or by pressing "CTRL+S".
23 Check the specification for problems via menu entry 2 of the ReqPat toolbar (cf. Figure 5 in Section 3.1).
24 The problems view shows a list of problems as in Figure 14 in Section 3.3.
25 Double-click on one of the red marked entries in the problem view. The corresponding entry in the requirements editor is selected.
26 Correct the used information to "unlockRequest".

27 Perform checking the document (cf. step 23). There are no problems listed anymore.

28 Create a further pattern-based requirement on the same hierarchy level than the entry created in step 15 (cf. step 7).

29 Open the ReqPat editor window, call content-assist and choose entry 1 (cf. (cf. Figure 5 in Section 3.1). Note that entry 1 corresponds to a different requirement pattern than in Figure 5 since we are in the section "Inputs & Outputs" of the function "CentralLocking".

30 Edit "INFORMATIONLIST" and specify the function "CentralLocking" creates the information "unlockDoors".

31 After performing a check on the document (cf. step 23), the problem view shows that "unlockDoors" was not yet defined and that it has to be defined with the corresponding requirement pattern.

32 Go to the section 1.3 "Inputs & Outputs" of the top-level function "BodyControlModule". Create a new pattern-based requirement (cf. step 7 or 15) and use the pattern as proposed by the problems view. Specify that the function "BodyControlModule" sends the information "unlockDoors" (see also Figure 1 in Section 2.1).

33 Check again the document (cf. step 23). There are no problems listed anymore.

34 Feel free to define further functions and input/output information according to the specification methodology sketched in Section 2.1.