Aristotle University of Thessaloniki

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CSSP Properties Specification & Verification Framework - User Manual

July 2016

Version 0.20

Aristotle University of Thessaloniki, Greece

École Polytechnique Fédérale de Lausanne, Switzerland

Document Revisions

Date	Version Number	Document Changes
07/06/2016	0.1	Initial Draft

Table of Contents

1 Introduction

1.1 Scope and Purpose

This document is the User Manual for the CSSP Properties Specification & Verification Framework that was introduced by the Aristotle University of Thessaloniki (GR) and the École Polytechnique Fédérale de Lausanne (Ch) during the "Catalogue of System and Software Properties" project, funded by the European Space Agency. The document constitutes the deliverable D11 of the project.

The scope of the CSSP Properties Specification & Verification Framework (PSV) is the requirements engineering processes carried out during the system software design of critical systems, aiming to support a correctness by construction methodology for:

- the systematic specification of requirements based on a knowledge base and a set of boilerplates, which are parameterized to suit the particular specification context;
- tracing the decomposition of requirements, the properties derivation and the requirements coverage checks;
- the systematic (re-)use of validated solutions, associated with property patterns.

An essential part of the PSV framework is the CSSP ontology, a knowledge base that contains all the concepts, attributes and relationships used to represent the system under specification, its functionalities and its architecture and, on top of this, a set of requirements and properties related to that system.

The tools of the PSV framework are:

i. the TopBraid Composer ontology editor;
ii. the BIP (Behavior, Interaction, Priority) tools for model-based design;
iii. the new integrated CSSP toolset, a GUI-based environment with functions for,
    a. the ontology-based requirements specification, properties derivation, searching and validation (CSSP tool),
    b. the ontology-based management of the specification context for applying existing solutions that enforce properties by correct-by-construction model transformations (AM tool);
iv. the DFinder tool for deadlock checking, and the nuXmv tool for model checking.

The integrated CSSP toolset is a new tool, whereas all the others are already existing tools, which are used within the PSV framework.

Multiple user roles are foreseen for the PSV framework with the following experience:

- the Ontology Engineer, who performs tasks related to the development and maintenance of the CSSP ontology for a particular domain; he is expected to be an experienced user of the TopBraid ontology editor;
- the Requirements Engineer, who uses the integrated CSSP toolset;

- the Verification/Validation Engineer, who uses the integrated CSSP toolset in combination with the DFinder and nuXmv tools;
- the System Software Engineer, who uses the integrated CSSP toolset in combination with the BIP tools.

The User Manual is focused on the functions of the integrated CSSP toolset and their interactions with the other tools of the PSV framework. For a detailed guide to the functionality of the other tools, the reader can refer to their user manuals [?].

1.2 Process Overview

Figure 1 presents the CSSP process and user role activities performed with the PSV framework. The shown activities are briefly described, in order to later introduce the various workflows based on the combined use of the tools of the PSV framework.

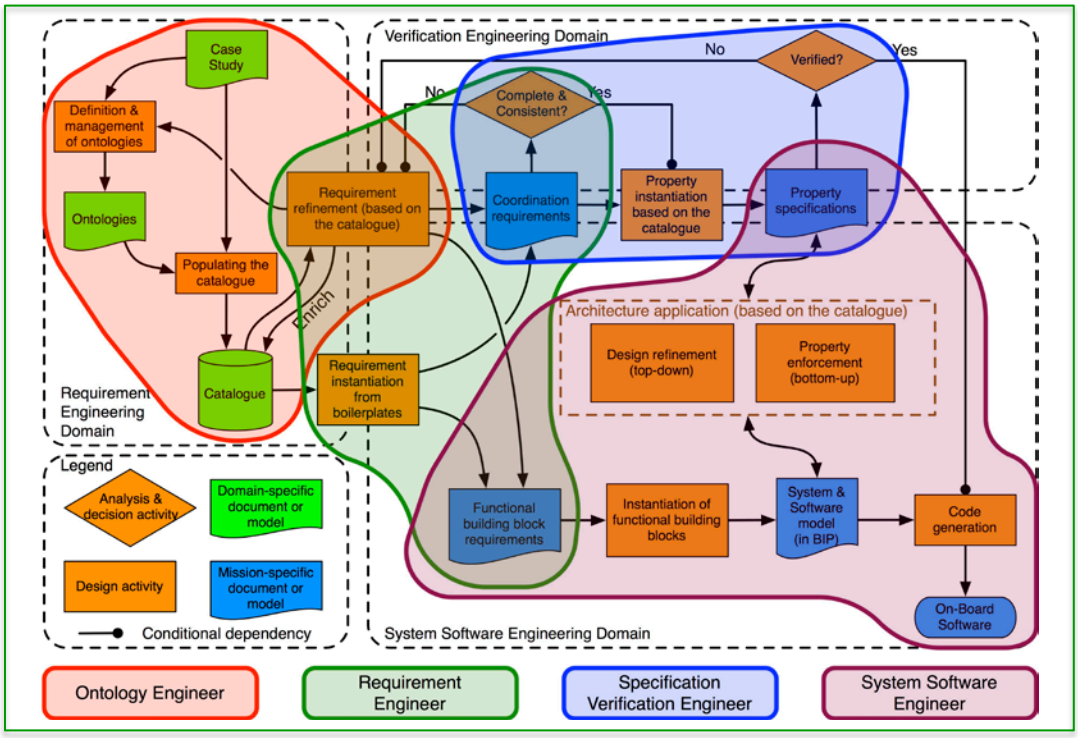

Figure 1: The CSSP process and user role activities

The CSSP ontology is first defined and the catalogue of requirements and properties is populated as it is shown in the top-left "Requirements Engineering Domain" box. The catalogue is a searchable ontology-based repository, collecting patterns for all of the requirements, properties and architectures (design solutions) to be used. These two activities can be repeated when a system of a previously unforeseen class is to be designed, whereas the catalogue can be enriched at any time, when new boilerplates are identified. The tools used for these activities are the TopBraid ontology editor and the integrated CSSP toolset (see Figure 2), which accesses the CSSP ontology through an underlying ontology server (Jena library).

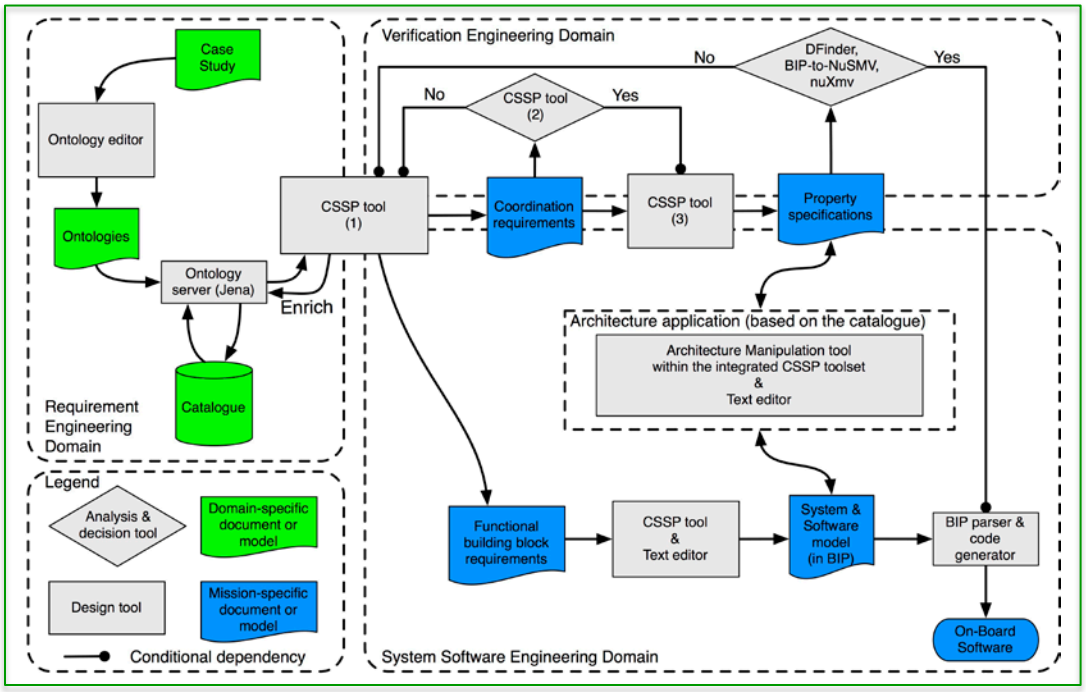

Figure 2: Tools of the PSV framework

The rest of the activities comprising the CSSP process are roughly grouped into the following steps, which are performed iteratively, refining the model of the designed system until the properties are satisfied and all the requirements are discharged:

1. System and software are instantiated from boilerplates available in the catalogue (Requirement instantiation and Requirement refinement activities in Figure 1 using the integrated CSSP toolset).
2. Completeness and consistency of the instantiated requirements is validated by querying the CSSP ontology and by analyzing the formalized representation of the requirements (Completeness & consistency check activity in Figure 1 using the integrated CSSP toolset).
3. Functional building blocks are derived from the corresponding requirements (Instantiation of functional building blocks activity in Figure 1 using the integrated CSSP toolset and a text editor for the BIP system model).
4. Properties are instantiated from the property patterns associated to the involved boilerplates (Property instantiation activity in Figure 1 using the integrated CSSP toolset).

5. Top-down design step: Architectures, i.e. formalized design solutions, corresponding in the catalogue to enforceable properties, are applied to the BIP system model (Design refinement part of the System software design activity in Figure 1 using the AM tool of the integrated CSSP toolset).
6. Bottom-up design step: Architectures from the catalogue are applied to functional building blocks, in order to build a set of additional enforced properties that will eventually entail those not discharged by the architectures in Step 5 (Property enforcement part of the System software design activity in Figure 1 using the AM tool of the integrated CSSP toolset).
7. Meet-in-the-middle: Additional verification is performed to ensure that:
    a. the composed system is free from deadlocks;
    b. the applied architectures are mutually non-interfering;
    c. all components satisfy typing requirements relevant to the operands of the corresponding architectures.

    Together, these three analyses ensure that all the enforceable properties entailed by the requirements are, indeed, satisfied (Property verification activity in Figure 1).
8. The BIP model is transformed into the input format of the nuXmv model checker, which is used to check the satisfaction of the properties that cannot be enforced by architectures (Property verification activity in Figure 1 using the BIP-to-NuSMV and the nuXmv tools).

A more detailed description of the CSSP process and the mentioned activities is provided in the deliverable D6 of the “Catalogue of System and Software Properties” project [?].

2 PSV framework system requirements and installation

2.1 Integrated CSSP toolset

The integrated CSSP toolset runs within a Java Runtime Environment 8, which supports JavaFx 2+. It is a standalone application with a JavaFx Graphical User Interface using the Apache Jena framework to access the CSSP ontology.

No particular installation procedure is required:

- On MS Windows workstations just double-click the executable file CSSP-Tool.exe.
- In Unix systems, the file CSSP-Tool.sh should run.

[Provide a concise description of the context for this sub-process or workflow, including any requirements or conditions that are relevant.]

[typical sequence for using the software to manage {group of workflows/functions}:

1. Configure your workspace
2. Manage a key workflow
3. Manage another key workflow
4. Report
5. Troubleshoot]

2.2 TopBraid Composer (Free Edition)

A free license for the TopBraid Composer can be obtained by selecting the product Free edition from the corresponding pull-down menu of the installation web page:

http://www.topquadrant.com/downloads/topbraid-composer-install/

The TopBraid Composer is implemented as an Eclipse plug-in. The system requirements for TopBraid Composer are the same as for the Eclipse 4.3 platform:

http://www.eclipse.org/eclipse/development/readme_eclipse_4.3.php

The installation of the TopBraid Composer is straightforward:

- For MS Windows workstations, first decompress the downloaded file. The tool can be installed in any folder with write access, but the Program Files folder is not recommended, because some Windows versions restrict the access to this folder. To run the software, double-click the executable file named TopBraid Composer.exe.
- In Unix systems . . . . . .

2.3 BIP tools

[Provide a concise description of the context for this sub-process or workflow, including any requirements or conditions that are relevant.]

2.4 DFinder

[Provide a concise description of the context for this sub-process or workflow, including any requirements or conditions that are relevant.]

2.5 nuXmv

[Provide a concise description of the context for this sub-process or workflow, including any requirements or conditions that are relevant.]

3 Ontology editing with the TopBraid Composer

3.1 General description

The TopBraid Composer is an enterprise-class modelling environment, fully compliant with the World Wide Web Consortium (W3C) standards, appropriate for developing Semantic Web ontologies and for building semantic applications. It is widely used to develop ontology models, to convert data and models in/from an RDF/Web Ontology Language (OWL) representation, to transform and support data source integration, and to develop Semantic Web services and applications.

TopBraid Composer is a complete editor for RDF Schema and OWL models, as well as a platform for other RDF-based components and services. It can load and save any Version 2 OWL file in formats such as RDF/XML or Turtle, and it supports various reasoning and consistency checking mechanisms based on a built-in OWL inference engine, the SPARQL query engine and the Rules engine. The OWL description logic (DL) is also supported via a range of built-in OWL DL engines such as OWLIM, Jena and Pellet.

TopBraid Composer also supports the SPARQL Inference Notation (SPIN), which can also be used to define integrity constraints that can highlight invalid data while editing the input knowledge. Finally, the TopBraid Composer also provides inference explanation facilities for Pellet and SPIN.

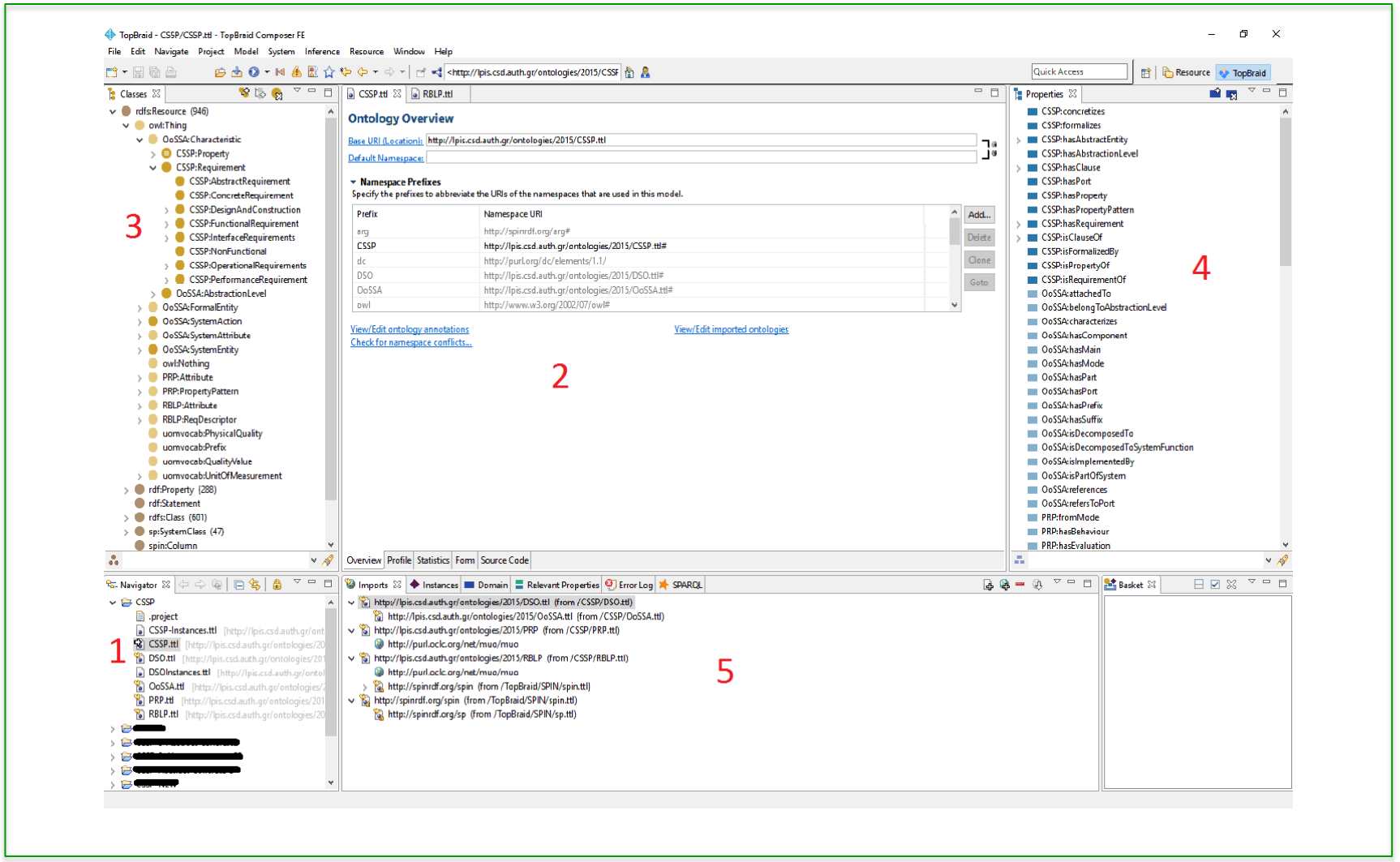

Figure 3: The TopBraid Composer graphical user interface

3.2 Graphical User Interface

The ontology editing can take place through TopBraid composer. In Figure 3, the TopBraid environment is shown:

- Navigator panel (1)

  This panel shows the existing ontology projects. Upon selecting a specific project, it is possible to view its files (sub-ontologies). For the CSSP ontology, these files are: `CSSP.ttl`, `OoSSA.ttl`, `RBLP.ttl`, `PRP.ttl`, `DSO.ttl`, `DSOInstances.ttl`, `CSSP-Instances.ttl`. The user selects any of these files by double-clicking on it, which causes its details to appear in Panel 2.

- Panel 2

  This panel shows the details for a selected entity. In the example of Figure 3, the details for the `CSSP.ttl` ontology file are shown. In this panel, it is possible to update existing entities, like for example adding new restrictions to a class.

- Classes panel (3)

  This panel presents the class hierarchy within the selected ontology file. By double-clicking on a class, its details appear in Panel 2. The Class panel also supports the creation of new classes. For this purpose, the user will have to right-click on an existing class, which allows him to create either a subclass or a sibling class of it. Even if no new classes have been created, the `OWL:Thing` class is already there. The update of a selected class takes place in Panel 2.

- Properties panel (4)

  This panel presents the OWL properties hierarchy for the selected ontology file. By double-clicking on a property, its details appear in Panel 2. The Properties panel also supports the creation of new properties. For this purpose, the user will have either to right-click on an existing property, in order to create a sub-property of it, or else to click the “create property” button on the top right part of the panel, in order to create a new property. The update of a selected property takes place in Panel 2.

- Panel 5

  In this panel, the following options are available for the user:

  - to view the imported files for a selected ontology,
  - to view the instances of a selected class,
  - to view the properties, for which a selected class is the domain,
  - to view the relevant properties for a selected class,
  - to view the error log,
  - to run SPARQL queries.

4 The integrated CSSP toolset

4.1 General description

The integrated CSSP toolset is a GUI-based application for ontology-based requirements specification and properties derivation (CSSP tool) that supports correct-by-construction model transformations (AM tool) for applying existing solutions to enforce the derived properties.

4.2 Graphical User Interface

From the Main Application screen shown in Figure 4 the user can (i) browse the categories of the catalogue of requirements, (ii) search for and find a particular requirement, and (iii) create a new or edit an existing requirement.

If a panel is not in use, it can be minimized by clicking the blue "-" (minus) button at its top right. The button is then replaced by a "+" (plus) button that restores the original size of the minimized panel.

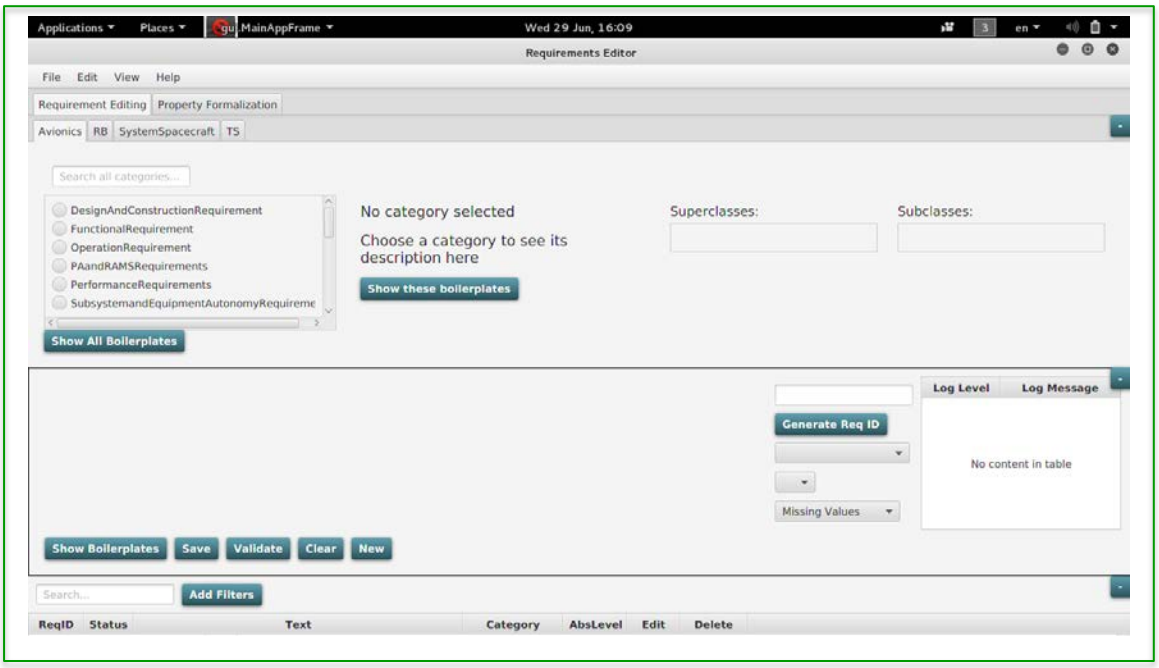

Figure 4: The Main CSSP Application screen

- Catalogue browser panel

  Figure 5 shows the Catalogue browser panel with all the other panels minimized. A list of selectable category names is displayed on the left, showing the top-level categories in the focused abstraction level tab. The selected category is shown at the middle, with its description and two lists on the right that are populated with the parents and children categories in the Catalogue

hierarchy. With the "Show All Boilerplates" button, the focus is changed to the Boilerplate selection panel. The same happens with the "Show these Boilerplates" button that highlights the most commonly used boilerplates.

The selected category does not restrict the scope of the search functions, which access all abstraction levels of the Catalogue.

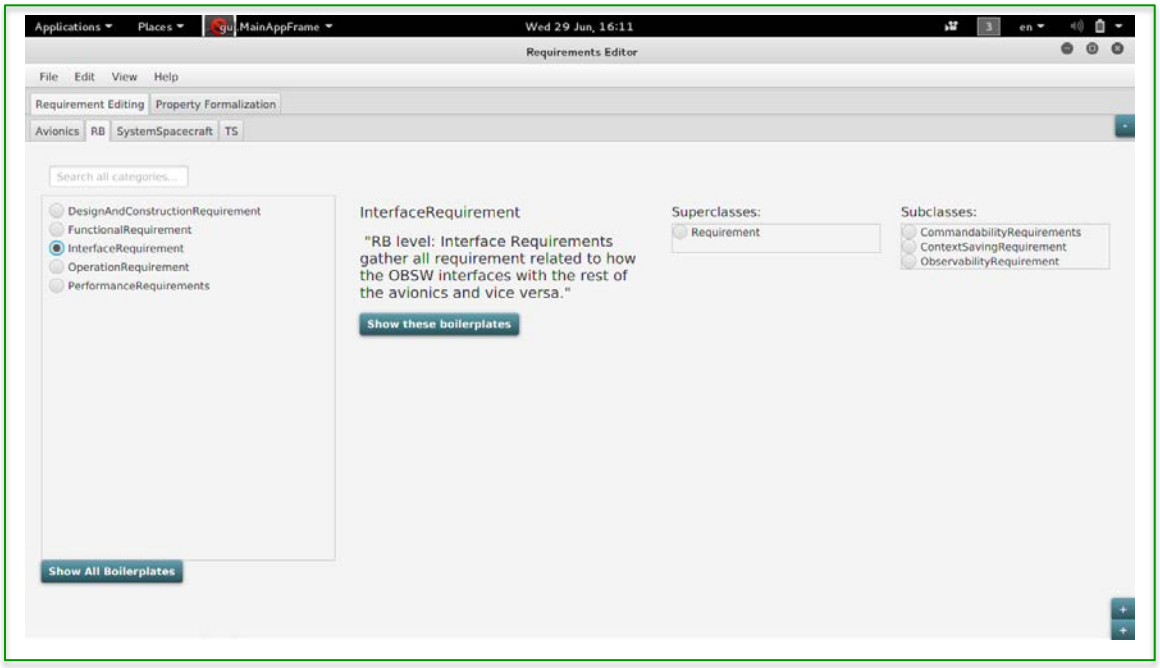

Figure 5: The Catalogue browser panel

- Boilerplate selection panel

  Figure 6 shows the Boilerplate selection panel with all the other panels minimized. All the prefixes, main body and suffix boilerplates are displayed, as they are retrieved from the input text file `SyntaxText.txt`, located at the top level application folder.

- Requirement editing Panel

  Figure 7 shows the Requirement editing panel with all the other panels minimized. The main body boilerplate of the requirement is displayed on the left along with its prefix and suffix boilerplates, if any. Boilerplate fields may be either a static string (like "shall" in this example), or a type-content pair, where the type corresponds to an ontology class and the content consists of ontology instances (e.g. "System Entity: The SMU" in Figure 7). These fields are color-highlighted with the green used for common words[1] with no ontological representation (like "for"), blue for words corresponding to ontology instances, purple for multiple words corresponding to a single ontology instance[2] (requires quotes) and red for words that have not been matched. These items

[1] Loaded from the file `PrepWords.txt` located at the top level application folder.
[2] Ontology entities consisting of multiple words (purple) are written in quotes, e.g. "very long entity".

can be edited; upon click, an editable text field is displayed. When the focus is moved elsewhere, their color coded labels reappear.

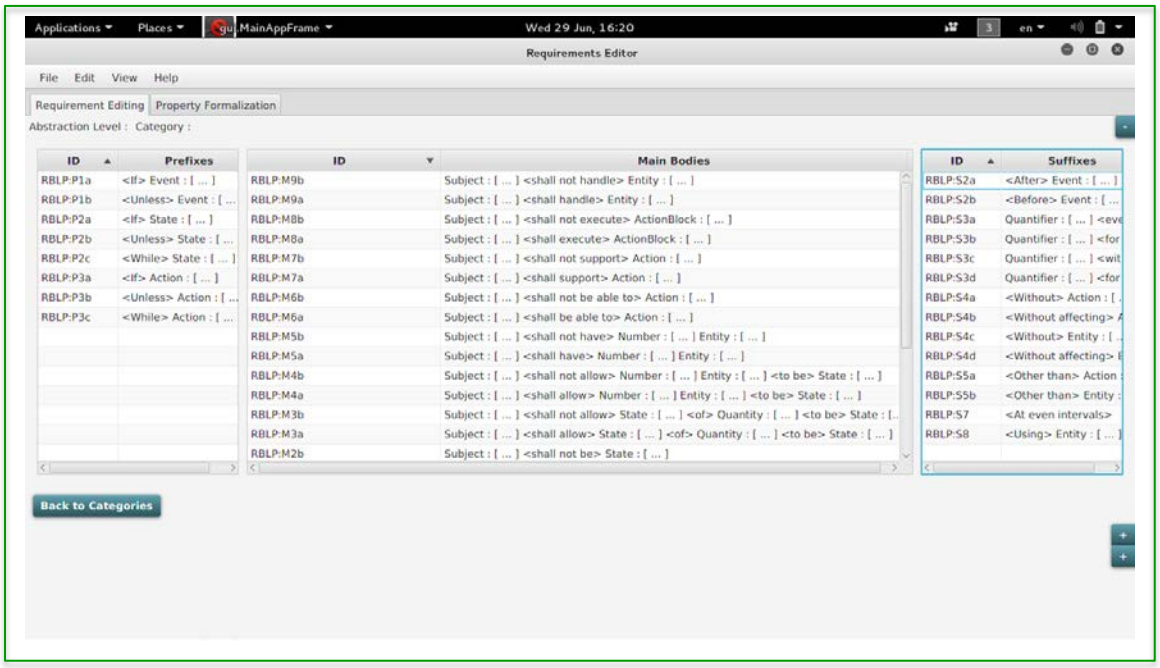

Figure 6: The Boilerplate selection panel

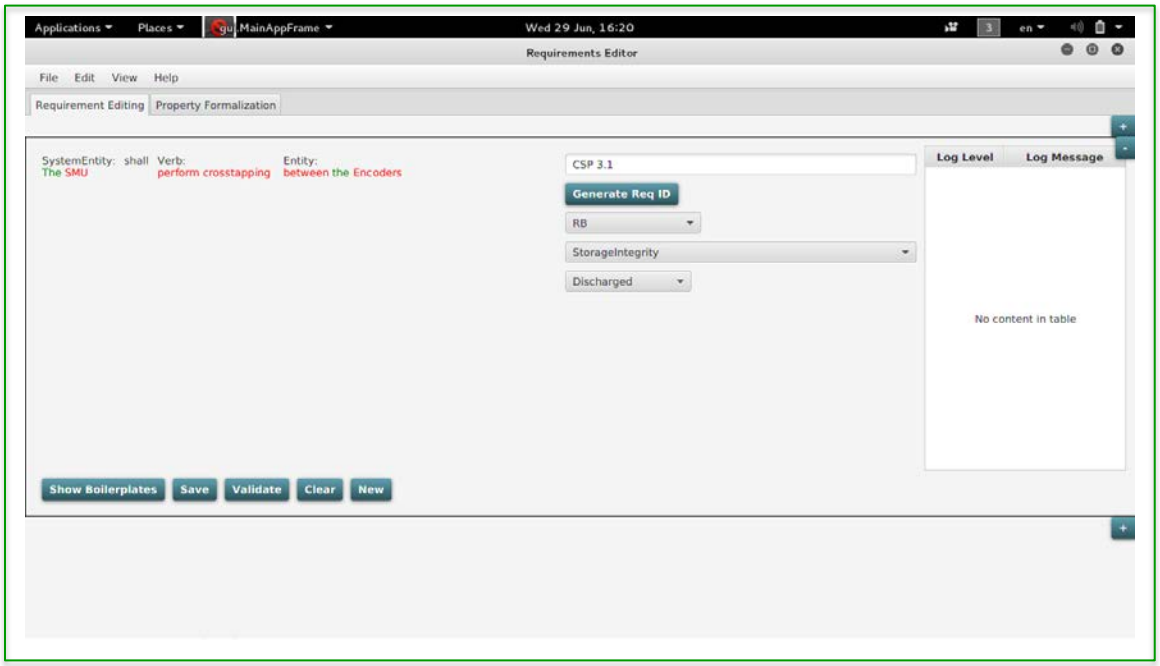

Figure 7: The Requirement editing panel

At the center of the window, edit functions are provided for modifying the requirement ID, its abstraction level and category, as well as its specification status. The console at the right displays diagnostic messages from the applicable validation checks. Upon clicking a message connected to a specific field, the relevant field is highlighted.

The button console underneath provides access to the following functions:

1. Show Boilerplates
   Displays the Boilerplate selection panel at the top of the Main Application screen.
2. Save
   Validates and saves the requirement. No changes are saved while editing until the button is pressed.
3. Validate
   The requirement is checked for errors.
4. Clear
   The Requirement editing panel is cleared. All changes are lost, unless the user has previously saved the requirement.
5. New
   A new requirement is created of the selected category and abstraction level. All previous changes to the edited requirement are lost, unless the user has previously saved it.

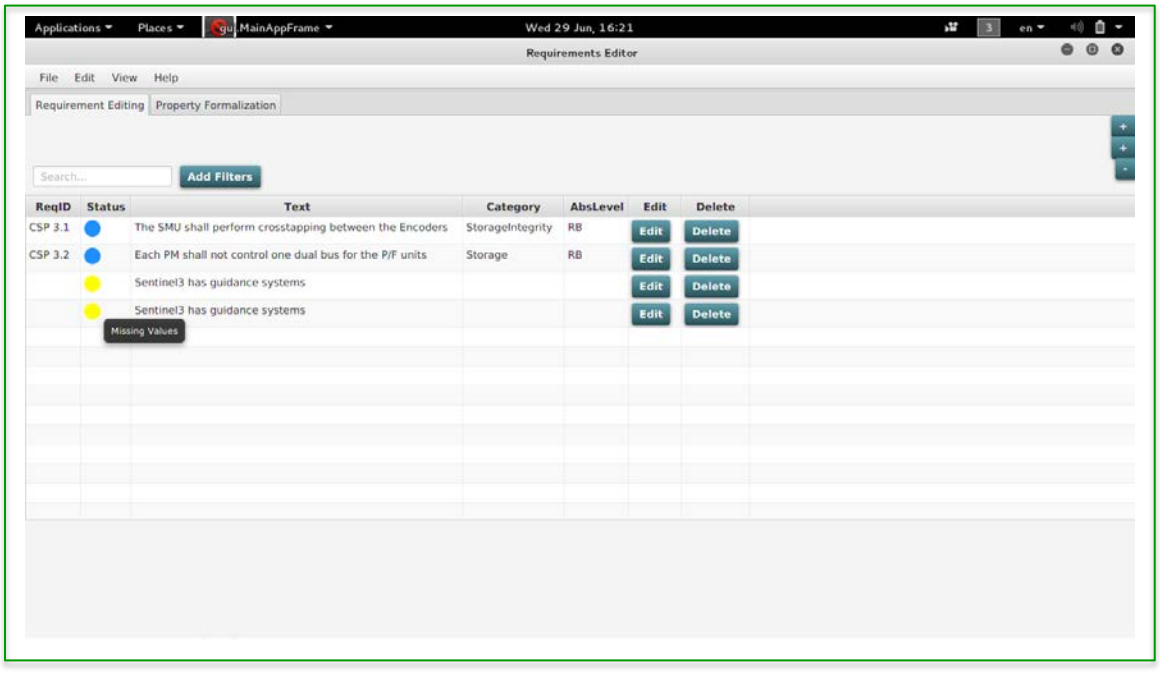

Figure 8: The Memory browser panel

- Memory browser panel

Figure 8 shows the Memory browser panel with all the other panels minimized. This panel displays all requirements that are currently loaded into the memory. Upon clicking the edit button in any row, the shown requirement will be displayed in the Requirement editing panel. Upon clicking any column header the items in the column will be sorted. The search field will match the given content against any field, such as the requirement ID, the status and any substring of the requirement's text, including the prefix and suffix that are not displayed in the browser.

4.3 Create a new requirement

4.3.1 Select the Abstraction Level and Catalogue Category

After starting the application, focus on the top part of the CSSP Tool. Select the tab that represents the Abstraction Level you want to search. To select a category, you can either use the search box in the top left, or you can navigate in the category hierarchy, using the buttons to the right of your screen to move to superclasses or subclasses of the category you have currently selected. Once you have found the category you wish to work with, click the "Show Boilerplates".

4.3.2 Choose the boilerplates to be instantiated

You will be taken to a new panel, with 3 distinct lists: Prefixes, Main Bodies and Suffixes. Those are the available boilerplates for the category you have selected. Adding any item will add it to the editing panel, in the middle of the screen, where you can further edit it. If you have previously selected any prefixes or suffixes for editing that are incompatible with the newly selected main body boilerplate, they will be cleared from the editing panel. Once you have chosen a main body and optionally, prefix and suffix, you are ready to begin editing.

Tip: Selecting a main body first is suggested; it will likely limit the available prefixes and suffixes that you can choose from.

4.3.3 Specify the requirement

The requirement editing panel is located in the center of the application. Here you can edit the body of a requirement as well as prefix and suffix (if they exist). Each of those 3 is comprised of lower level items, that you can view and edit. Those items have a type, which is displayed as a title and a content, which is displayed as a color-coded text string. In the color coded scheme, blue represents a single-word item found in the ontology, purple represents a multi-word item found in the ontology, green represents a connecting word item found in the ontology and red represents items not found. Quotes can be used to force the program to evaluate the contents of multiple words as one entity. A content assist system will provide suggestions of already existing instances of the given type to the user, as well as existing instances of subclasses of this type. The ID field is by default filled with a unique, automatically generated ID that is based on abstraction level and category. Using the

additional editing panel, the users can change properties of the requirement such as: Abstraction Level, Category, ID and Status.

4.3.4 Validate the requirement

Once you are done editing a requirement, you can have it validated by clicking the "Validate" button. Any errors discovered through the validation process are written to the requirement editing console, on the right of the panel. Clicking an error will highlight where that error occurred (if applicable). Finally, the tool will attempt to provide solutions if able in a new window.

4.3.5 Save the requirement

Once you have completed validation, you may proceed to saving the requirement. This adds the current requirement in the running memory, but does not save it in the ontology, until the user chooses to save to a file, or exits the application. An additional step of validation is ran when you click save, but the requirement will be saved regardless of any warning or errors that are discovered by validation.

4.4 [Sub-Process or Workflow Step 1] Example: Configure Connect to manage your work

[Provide a concise description of the context for this sub-process or workflow, including any requirements or conditions that are relevant.]

4.4.1 [Procedures for Step 1]

Example: To Log in to Connect:

1. Do something.
2. Complete an action.
3. Select a value.

Screen capture

4. Enter some text.
5. Drag and drop a value.

6. Click or press something to complete the procedure.

[NOTES, CAUTIONS, and WARNINGS provide any relevant or supplemental information about consequences of performing a step incorrectly. Place warnings before the step to be taken. Notes may be placed either before or after the corresponding step.]

NOTE: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam at porta est, et lobortis sem. Duis imperdiet in nisl sed luctus..

4.4.2 [Procedures for Step 2]:

1. Do something fairly complex that needs substeps:
   a) Do the first thing
   b) When something happens, do the next thing.
   c) Click a button or a link.
   d) Enter some data.
   e) Complete the action.
2. Complete an action using one of the following options:

- Option 1
  a) Click somewhere.
  b) Enter something.
- Option 2
  a) Click somewhere.
  b) Enter something.

3. Select a value.

Screen capture

4. Enter some text.
5. Click or press something to complete the procedure.

4.5 [Sub-Process or Workflow Step 2] Example: Asset Record Statuses

[If a sub-process or workflow step requires additional context and detailed information to properly prepare the user, you can expand a context section to include tables (such as definitions of the values available in a dropdown) or graphics such as flow diagrams. You should add captions to identify tables and graphics, and consider including a table of figures following the TOC if the number of graphics and tables is significant.]

Proin euismod lectus sed dui accumsan lobortis. Donec iaculis sed magna ac aliquam. Donec sagittis mi at enim gravida, vitae pharetra nunc sollicitudin. Suspendisse mollis turpis in odio lobortis tincidunt. Nullam ut augue eget massa eleifend consequat. Praesent ac vestibulum leo, sit amet tempor urna. Praesent eu quam diam. Morbi tincidunt nec urna at vehicula. Vestibulum tincidunt sit amet urna eget auctor. Nulla faucibus nulla vitae pretium rutrum. Nulla nibh sapien, ultricies eu pellentesque fermentum, molestie et purus.

Indicator	Definition
Value 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Value 2	Sed id neque auctor, pellentesque quam vel, pulvinar lorem.
Value 3	Morbi in odio vitae dui dictum ultricies eu vel nisi. Phasellus eu dui vitae nisl viverra vulputate ac sit amet turpis.
Value 4	Vivamus tristique augue ornare lorem lobortis, a pellentesque felis blandit.
Value 5	Duis at nisi eget ligula fermentum pretium at et felis. Integer consectetur nibh a condimentum rhoncus.

Figure X-X: Lorem ipsum dolor sit amet

If you encounter issues not addressed by this user guide, please contact your account manager for additional support.

5 Appendices

[Appendices are optional, and are used to provide additional detailed information that may help the end user manage the overall application. Examples could include references to standards (such as W3C standards), technical specifications required for regulatory compliance, checklists, or other information of a technical nature.]

6 Index

[Depending on the size or complexity of the final document, consider pulling together an index to assist the using in location specific information. Index entries correspond to tags or categories, and are useful in navigating long books.]

6.1 Style Sheet Information

The following styles have been set up in this template. Avoid applying manual character formatting to the document. Applying these styles will assist in the conversion process if the document is to be laid out in a structured authoring tool, content management tool, or an HTML editor.

Style Name	Apply to
Title	Title as listed on the cover page of the document
Subtitle	Subtitle as listed on the cover page of the document
7 Heading 1	Chapter Name or Process or Workflow
7.1 Heading 2	Subsection or SubProcess or Workflow step
7.1.1 Heading 3	Subsection 2 or Procedure
Callout Block Copy Note	Notes, cautions or warnings, use arrow graphic on the left margin
Chapter Body Copy	Generic text following a heading
• Chapter Body Copy – Bullet	Unordered list within a section or subsection, sometimes within a Step to indicate alternative ways to do something.
o Chapter Body Copy – Bullet 2	A secondary unordered list, within a higher level ordered or unordered list
6. Chapter Body Copy – Step	An ordered list (sequential) used in a procedure to indicate the order of actions to be taken
c) Chapter Body Copy – Step a	A secondary ordered list, e.g. substeps in a procedure
Chapter Body Copy – Indent	Sets additional text inward so that it aligns with either Chapter Body Copy – Bullet or Chapter Body Copy – Step
Chart Body Copy	Text within a table
Chart Header Information	The first row of a table.
Caption	Descriptive text for a table or graphic.
Header	Text that appears at the top of each page.

Style Name	Apply to
Footer	Text that appears at the bottom of each page.
Chart Title and Footer Info	Copyright and other front matter preceding the Table of Contents; additional information used to footnote or provide a legend for a table.
TOC Heading	Heading used for Table of Contents, Table of Figures (if applicable), and the Document Revisions pages of the front matter to the document.
TOC 1	TOC display information for a chapter. Generated automatically from Heading 1.
TOC 2	TOC display information for a sub-section. Generated automatically from Heading 2.
TOC 3	TOC display information for a subsection 2. Generated automatically from Heading 3.
[Template Instructions]	Guidance on building out the user guide. Should be deleted prior to publishing.