TTool

ttool.telecom-paris.fr

Use of TTool explained through the creation of a coffee machine.

	Document Manager	**Contributors**	**Checked by**
Name	Ludovic APVRILLE	Christophe Manseau, Philippe Paquette, Ludovic APVRILLE	Ludovic APVRILLE
Contact	ludovic.apvrille@telecom-paris.fr		
Date	July 7, 2021		

Contents

1 Preface

1.1 Table of Versions

Version	Date	Description & Rationale of Modifications	Sections Modified
1.0	July 7, 2018	First draft	
1.1	Oct 29, 2019	Adding safety pragmas	
1.2	Jul 2, 2021	Updating document with Sophie Coudert's remarks	

1.2 Table of References and Applicable Documents

Reference	Title & Edition	Author or Editor	Year

1.3 Acronyms and glossary

Term	Description

2 Introduction

TTool is a free and open-source toolkit dedicated to the design of embedded systems based on UML and SysML diagrams. The source files of TTool are available from its public git: `https://gitlab.telecom-paris.fr/mbe-tools/TTool`.

Installing TTool can be done either from the git, or by first downloading an installer from the website of TTool `https://ttool.telecom-paris.fr/installation.html`. This webpage also explains how to start TTool in Windows, MacOS, Linux.

3 Getting Started

This section assumes that you were able to start TTool.

3.1 Start new project

To create a new project, click on the "new" button as shown in Figure 1 below or select file and then new on the main tab. As shown in Figure 2, the main TTool window contains three different areas, the Project navigation window, the Design window and the Map view window; allowing the user to navigate through the files of the project and rapidly search for elements of a design, observe and create the SysML diagrams of the design, and have a bird's eye view of the current diagram respectively.

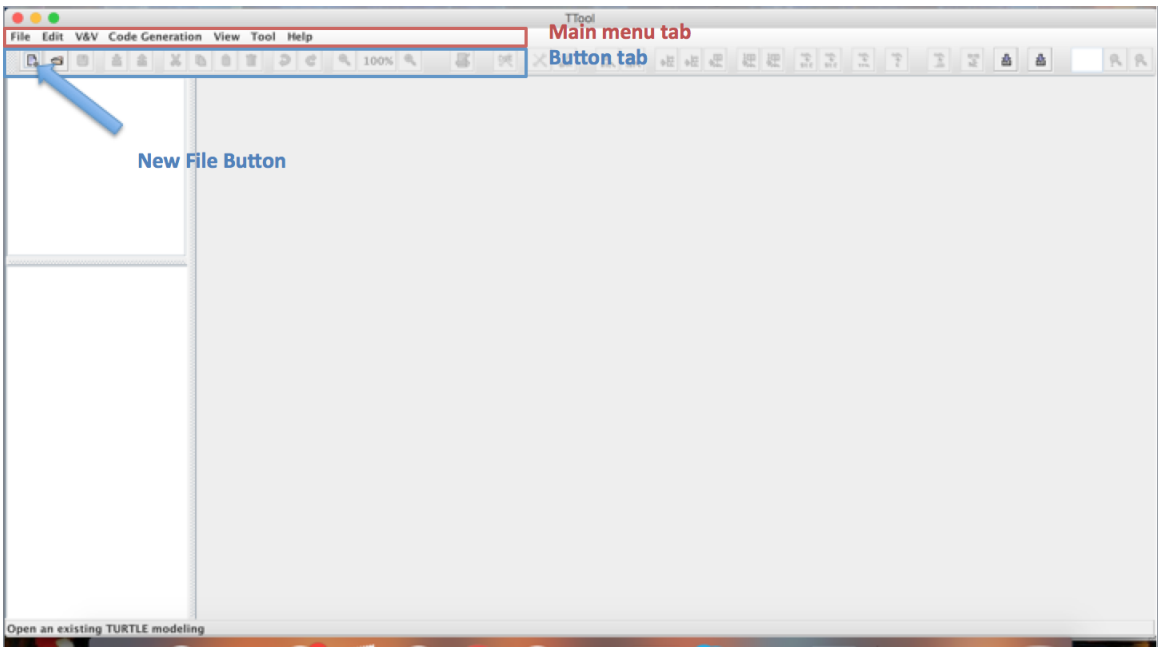

Figure 1: Create a file

To begin, we will start with the creation of a Methodology diagram as the one shown on Figure 3. In order to do this, right click on the design window and select "New AVATAR Methodology" (Figure 2). For each box on this diagram another diagram from the development of the project will be selected, however, this example will not include any properties or prototyping and therefore, these two boxes will remain blank. After all the other diagrams are created by following the steps in the remainder of this manual, one will go back to the

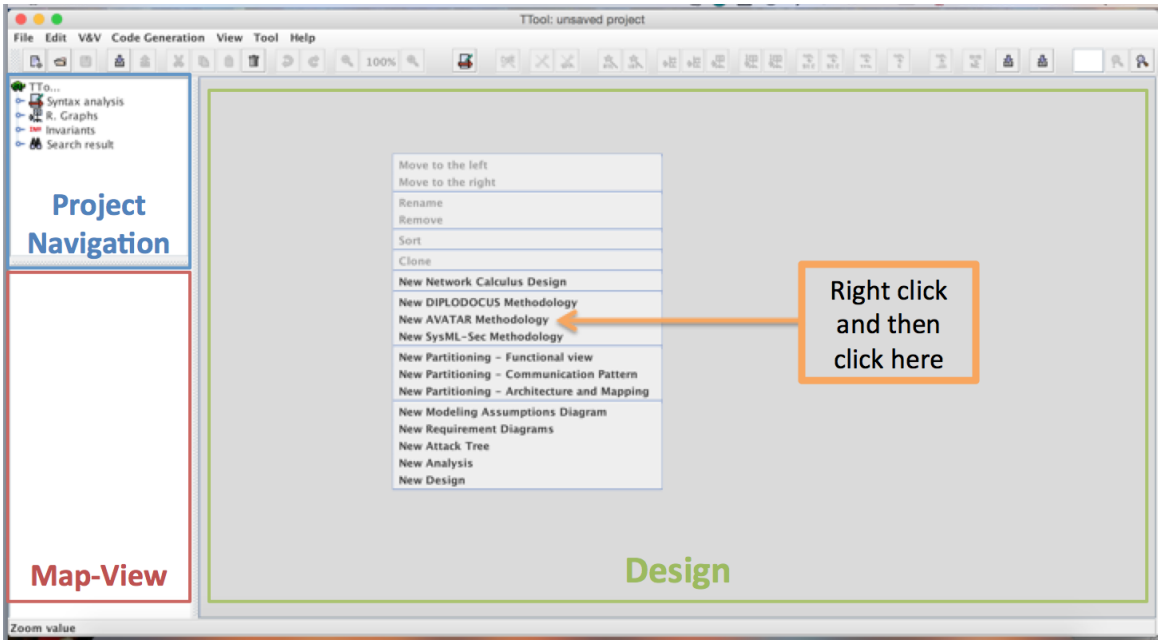

Figure 2: Create an AVATAR Methodology

methodology diagram, double click on each box, select the diagram corresponding to it and click on the arrow as shown in Figure 4 in order to select it.

4 Analysis

4.1 Assumptions

As shown in Figure 5, right click on the panel tab and add a "New Modeling Assumptions" diagram. By right clicking on the panel, it can be moved left or right. You can also drag-and-drop a panel to another position. You will now have an Assumptions panel and underneath it an Assumptions diagram tab. You can right click on this tab in order to rename it, in this example, we changed its name to 'System'. You will then add an assumption box by clicking on the AST button as shown in Figure 5. You can add as many assumptions as needed in order to fully explain what is being taken into account for the system.

1. Edit AVATAR Modeling Assumptions Diagram
2. Adds a comment: add a comment to the diagram
3. Comment connector
4. AST: adds an assumption box to the diagram
5. Diagram reference: add a box who refers to another diagram
6. Element reference: add a box who refers to an Avatar element
7. Composite: Splits up a compound assumptions into elementary ones
8. Versioning: classifies changes that are made to the original model

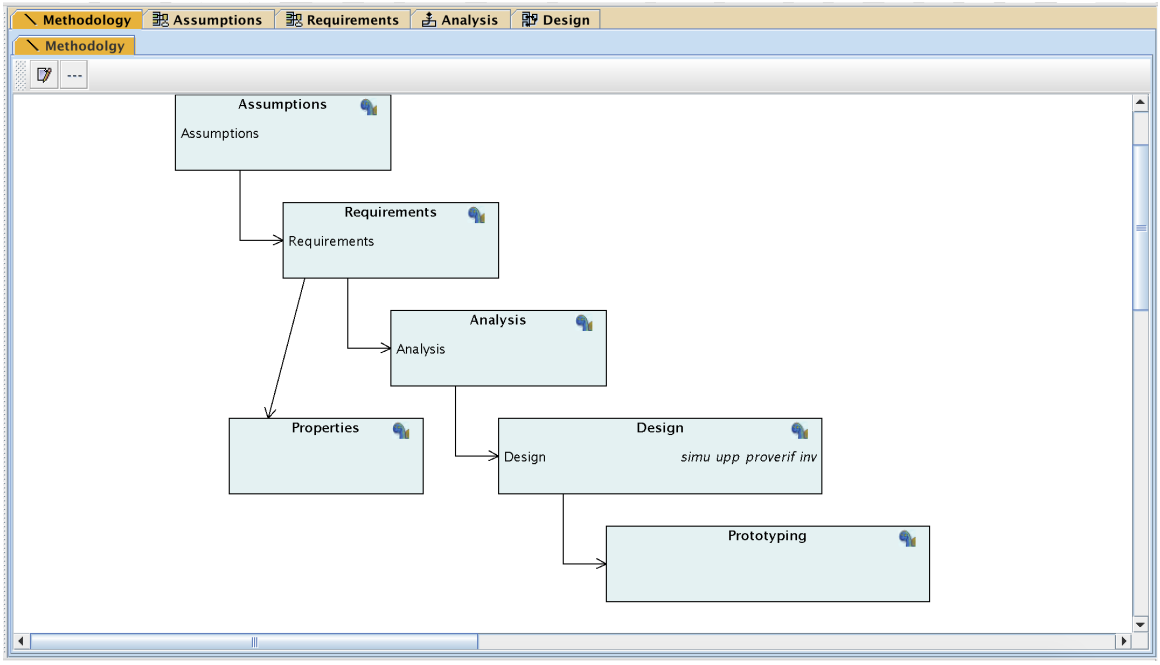

Figure 3: Methodology Diagram

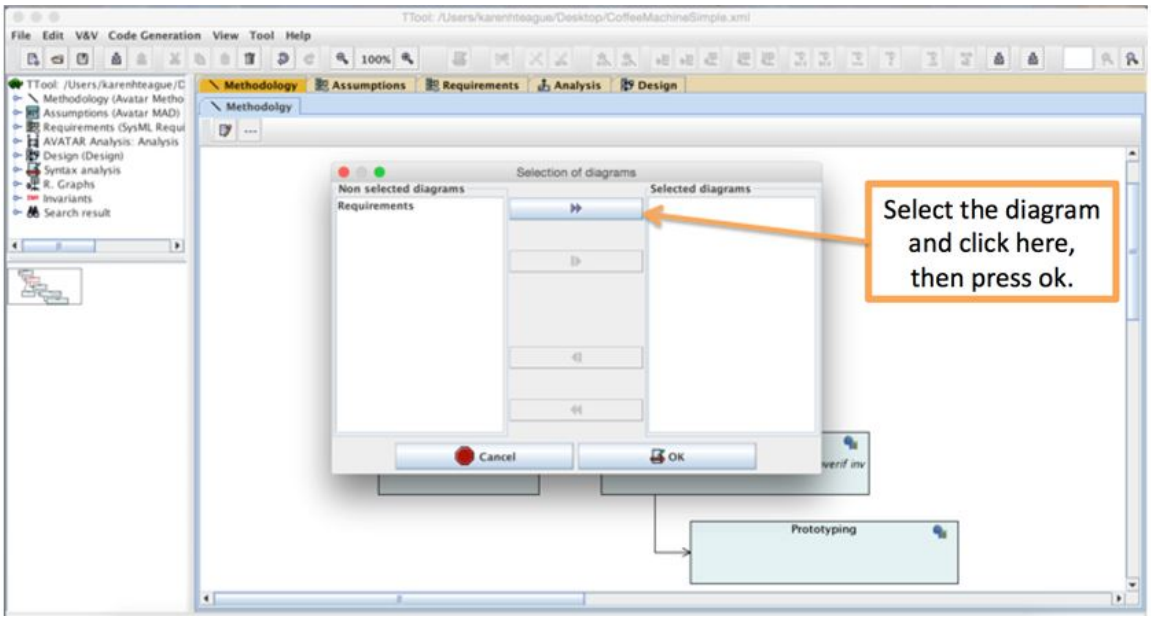

Figure 4: Selecting Diagrams

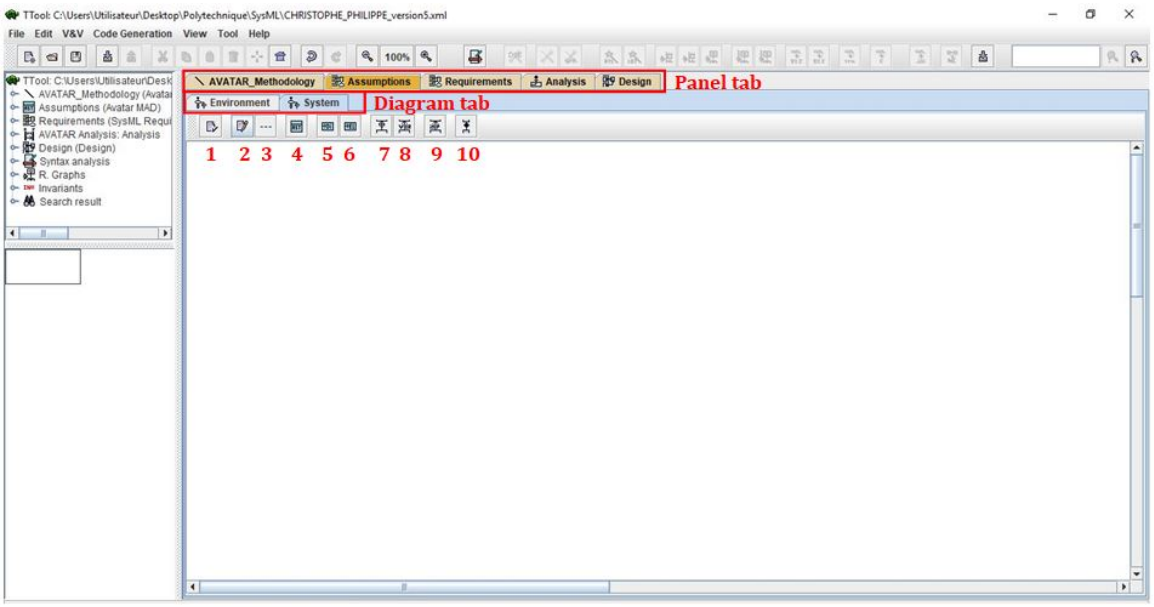

Figure 5: Assumptions Window

9. Impact: indicates that the assumption at the origin of the link as a direct impact on the referenced element at the end on the link.

10. Composition connector: relations between a diagram reference and elements references

To edit each of these boxes, one should double click on them. A window like the one shown in Figure 6 will appear. In this window one can modify the name and type of the assumption, as well as other attributes such as durability, source, status and scope. Also, a little description of the assumption can be added in the box on the lower left corner. Furthermore, different types of links, or connections, can be added between the different boxes by clicking on the respective button shown in Figure 5 and then on the perimeter of each of the boxes you wish to link.

Finally, in this example we have created two assumption diagrams; one for the system itself and one for the environment. The first one contains all the assumptions directly related to the system, while the second includes assumptions that may not be directly related to the system, but that may affect its performance as shown in Figure 8. One can add as many diagrams as one pleases by simply right clicking next to the existing diagram and selecting "New AVATAR Modeling Assumptions Diagram" and following the previously mentioned steps.

It is important to keep in mind that these assumptions will allow the creator and others needing to interact with the system to keep track of what was and was not considered during design. Therefore, one should attempt to be as descriptive as possible. This will make future improvements and changes easier while providing essential information about the system itself, resulting in a more complete design.

As shown in Figure 7, the system assumptions made for the coffee machine are those who enable the system to perform without malfunction. For example, we have assumed the lack of failures due to power outages and connectivity failures.

The environment assumptions were divided in two main categories: the sensors and the actuators. In this example, due to the short number of environment assumptions, they were

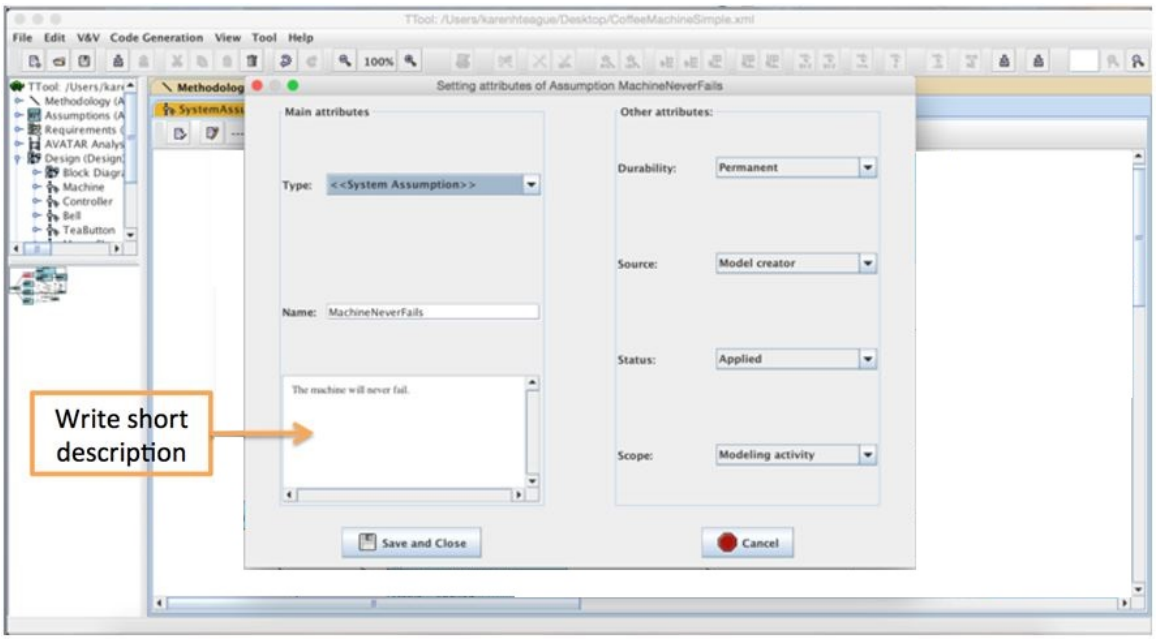

Figure 6: Assumptions Editing

all created on the same diagram as shown in Figure 8. Another option would be to user three diagrams for the environment assumptions. The first one would indicate the link between the environment and the two other diagrams, which would be the sensors diagram and the actuators diagram.

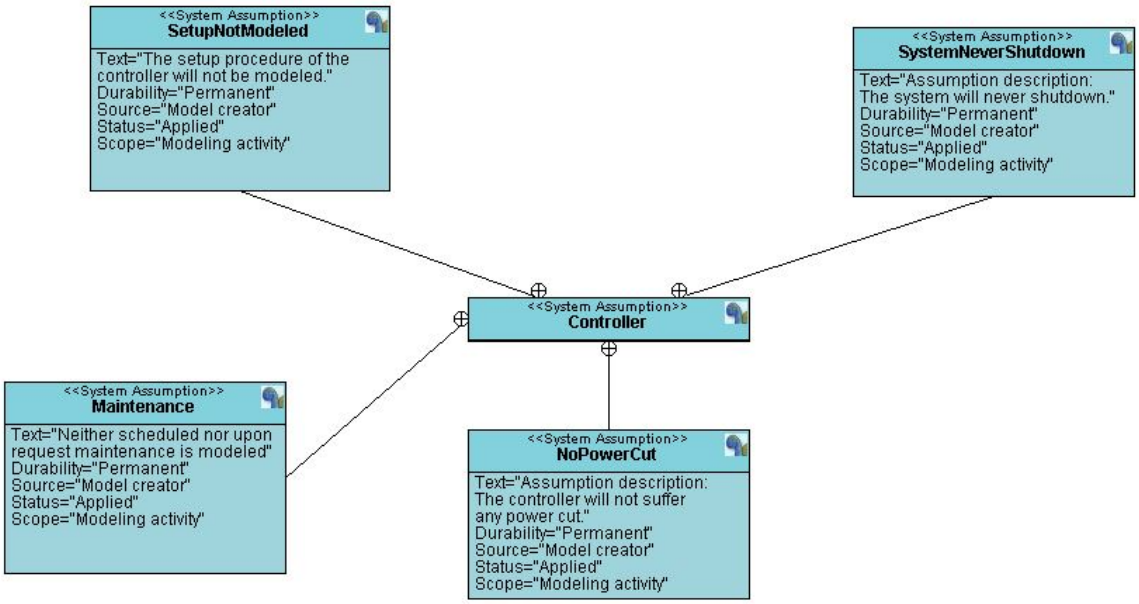

Figure 7: System Assumptions

The coffee machine has two type of sensors, a currency detector and pushbuttons. Together, they allow the users to order a cup of coffee or a cup of tea, and to pay for it. For their part, the actuators shall allow the users to receive their order. This part was fulfilled by the mechanical device. A mechanism permitting a transfer of information to the users was added to improve the ease of use of the coffee machine. As for the system, the assumptions

enable the system to perform without malfunction.

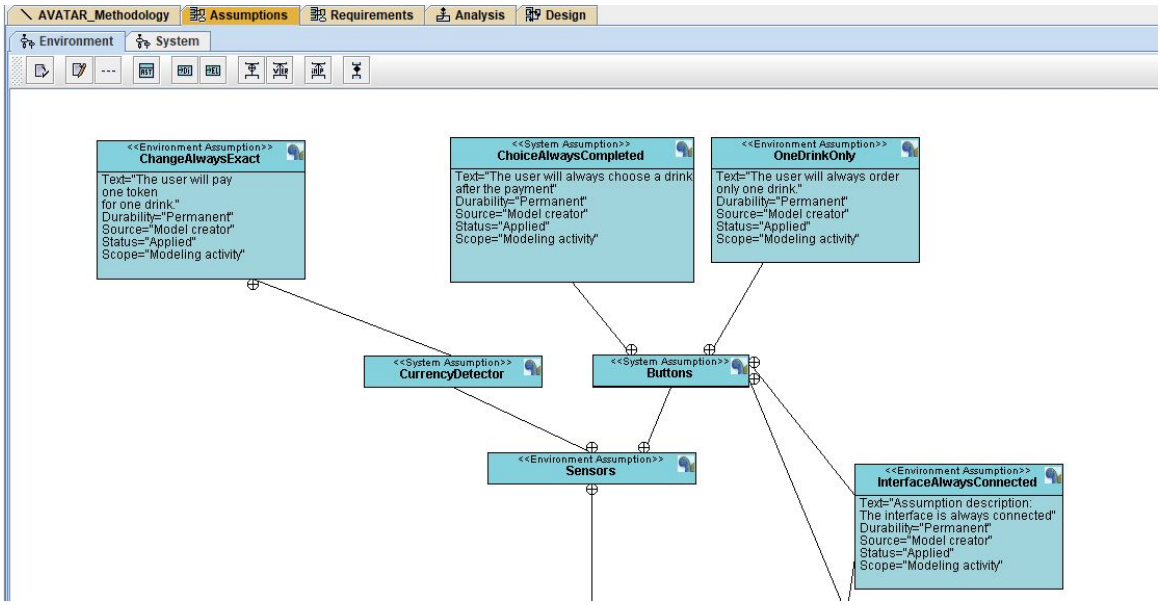

Figure 8: Environment Assumptions

4.2 Requirements

A requirement is a capability or a condition that the system must satisfy. It can be about a function that the system shall perform or a performance condition that must be achieved. Figure 9 shows the standard form and information of a requirement box.

Figure 9: Requirement Box

- Requirement name
- ID: unique identifier
- Text: text requirement
- Kind: status
- Risk: priority
- Reference elements

Use cases, which are part of analysis diagrams, can be useful to express functional requirements. However, they are no suitable for non-functional requirements. Requirements diagrams support introduce text-based requirements, providing a way to express both functional and non-functional requirements. Requirements diagrams are expected to be modeled after setting up the assumptions for our system.

We will create one or several requirement diagrams defining the requirements that the coffee machine controller will have. To do this, right click on the panel tab as before, and select "New Requirement Diagrams". Just as before, a new diagram tab will appear underneath. Once again, this diagram tab can be renamed as preferred following the same steps we used in the assumption diagrams.

For the coffee machine, we will divide the requirements in three parts: the requirements related to the inputs, the control and the outputs. The first diagram is named "General Requirement Diagram". It contains the link between the three parts mentioned above, which will all have their own requirements diagram.

Figure 10 shows the structural elements available to build the requirements diagram.

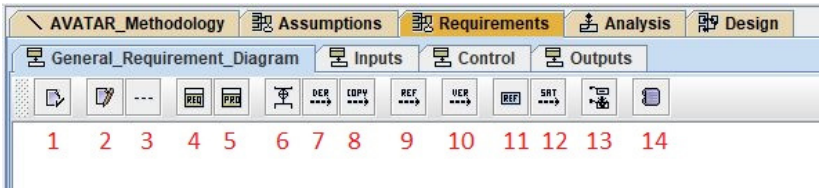

Figure 10: Structural elements of the requirements diagram

1. Edit Requirements Diagram
2. Add a comment: add a comment to the diagram
3. Comment connector
4. Requirement: add a requirement box to the diagram
5. Property: add a property box to the diagram
6. Composition: a relationship who defines the requirements hierarchy
7. DeriveReqt: a relationship who links two requirements of the same hierarchy but at different levels of abstraction
8. Copy: a relationship that connect a slave requirement a master requirement
9. Refine: a relationship who links used between a requirement that refine another
10. Verify: a relationship who defines how a model element verifies a requirement
11. Element reference
12. Satisfy: a relationship who show that a concept satisfies a requirement
13. Show/hide element attributes
14. Enhance

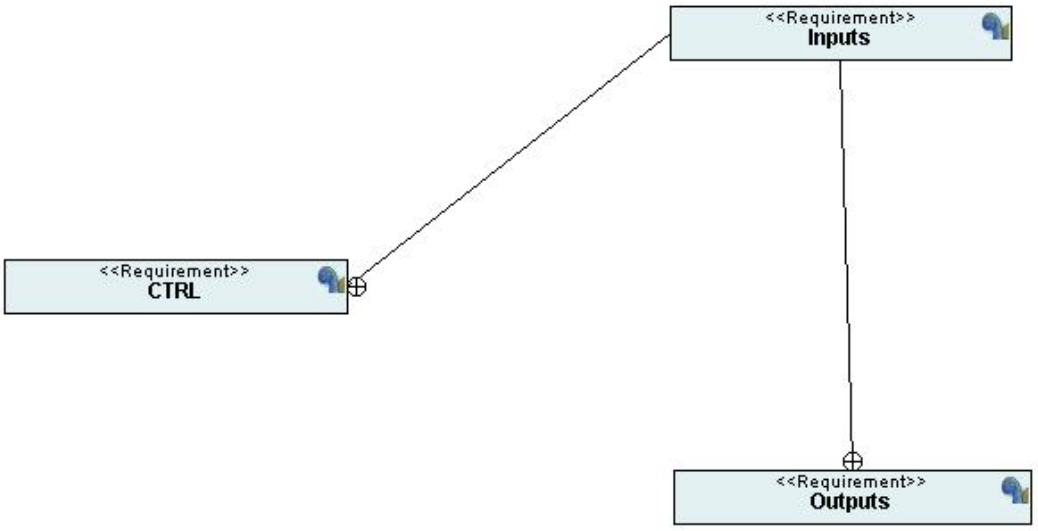

Figure 11: General Requirement Diagram

Figure 11 represents the general requirement diagram obtained for the coffee machine controller.

In a similar way to that of the assumptions, a new requirement can be added by selecting the requirement button shown in Figure 10. Once created, it can also be modified by double clicking on it. When the new window pops up, one can provide a description and change the ID, Type, Risk and Reference attributes of each requirement. The next step was to create the inputs diagram, the control diagram and the outputs diagram. By using the requirement box and relationships described in Figure 10, we obtained the diagrams shown in Figures 12 and 13.

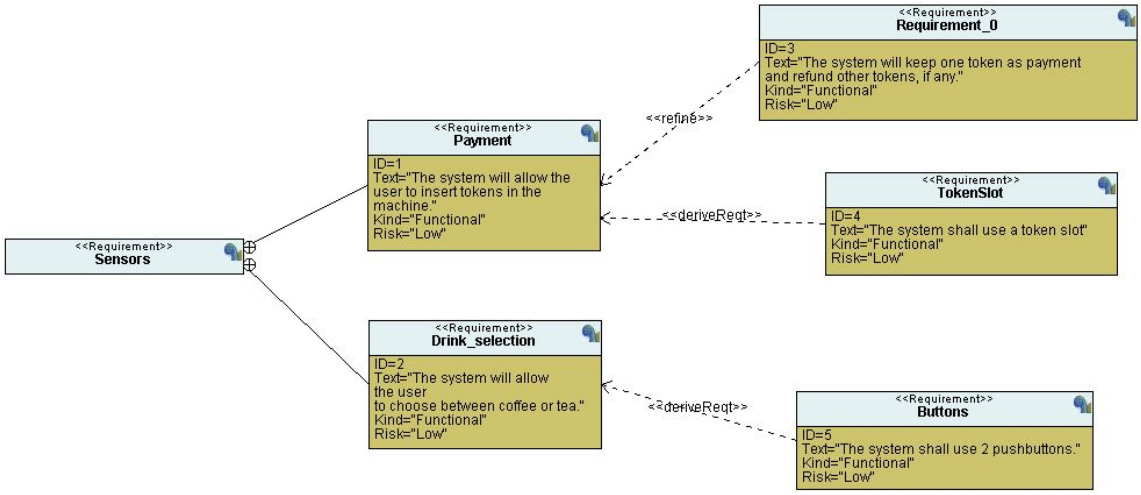

Figure 12: Inputs requirements diagram

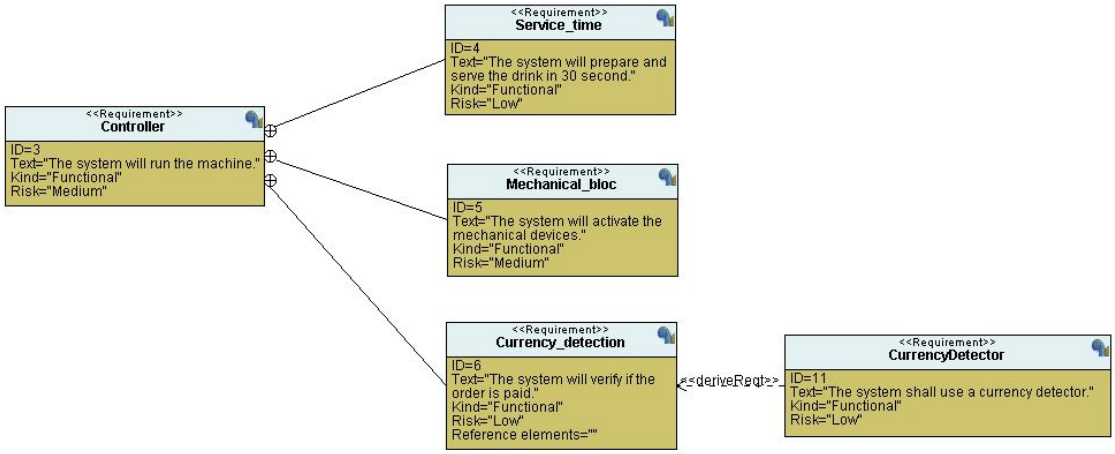

Figure 13: Control requirements diagram

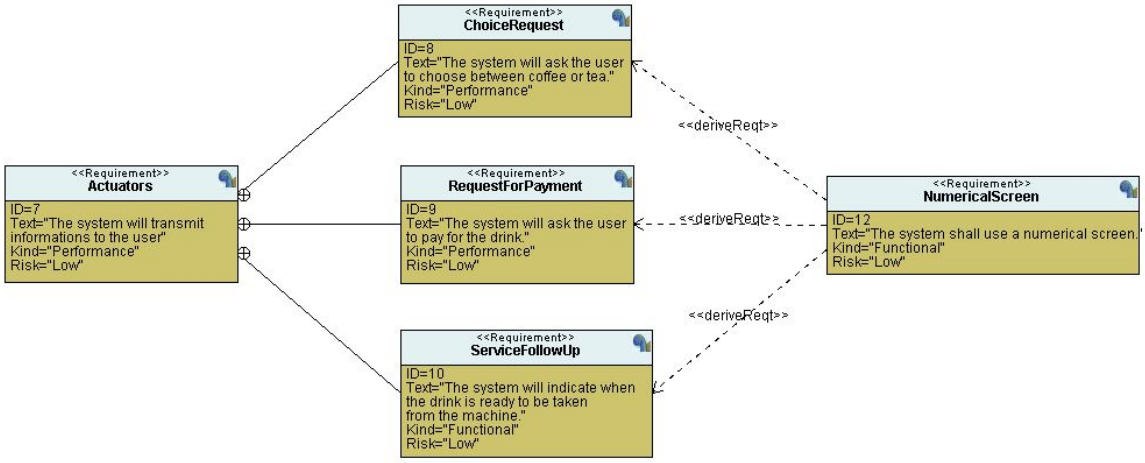

Figure 14: Output requirements diagram

4.3 Use Case

Once again, right click on the panel tab. This time, select the "New Analysis" option, which will give you a window like the one shown in Figure 15. We will begin the analysis process with a Use Case Diagram. To add it, right click on the window and select "New Use Case Diagram". This diagram will allow us to represent the system, the actors acting on it and the different instances or use cases that will be modeled. All these elements can be observed in Figure 17. To add each of them you should use the buttons shown in Figure 16. To change the names, double click on each on them.

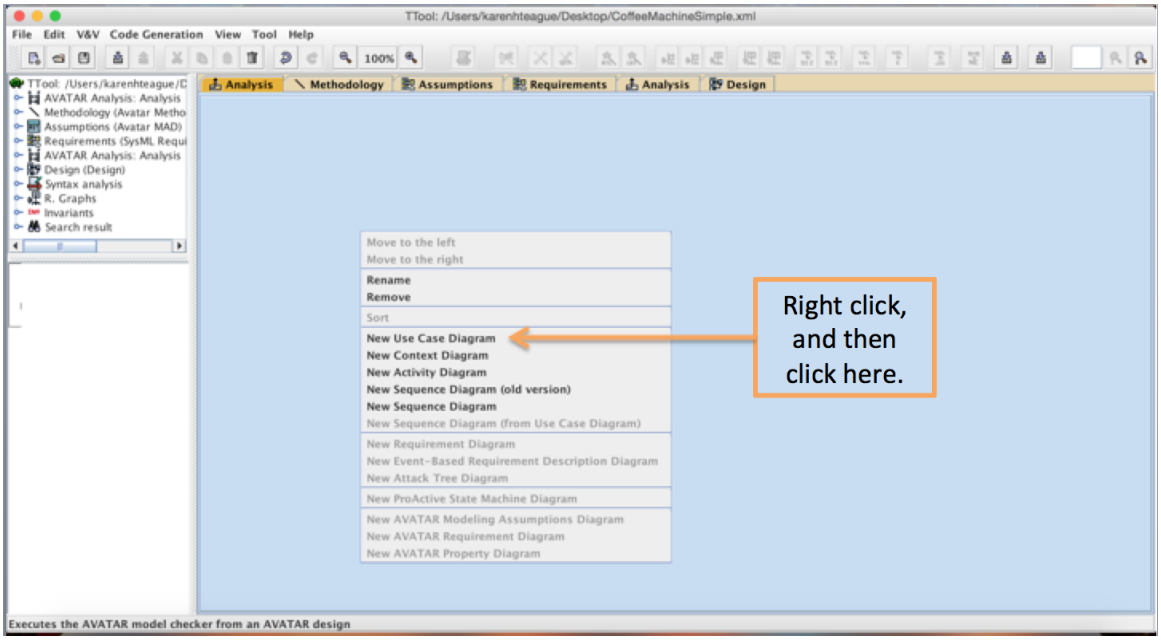

Figure 15: Create Use Case Diagram

Figure 16: Structural elements of the use case diagram

1. Edit Use Case Diagram
2. Add a comment: add a comment to the diagram
3. Comment connector
4. Add a border: create a system box who defines the boundaries of the system
5. Actor: add an actor to the diagram
6. Actor (Box Format): add an actor box to the diagram
7. Add a use case
8. Actor <->use case: links an actor to a use case

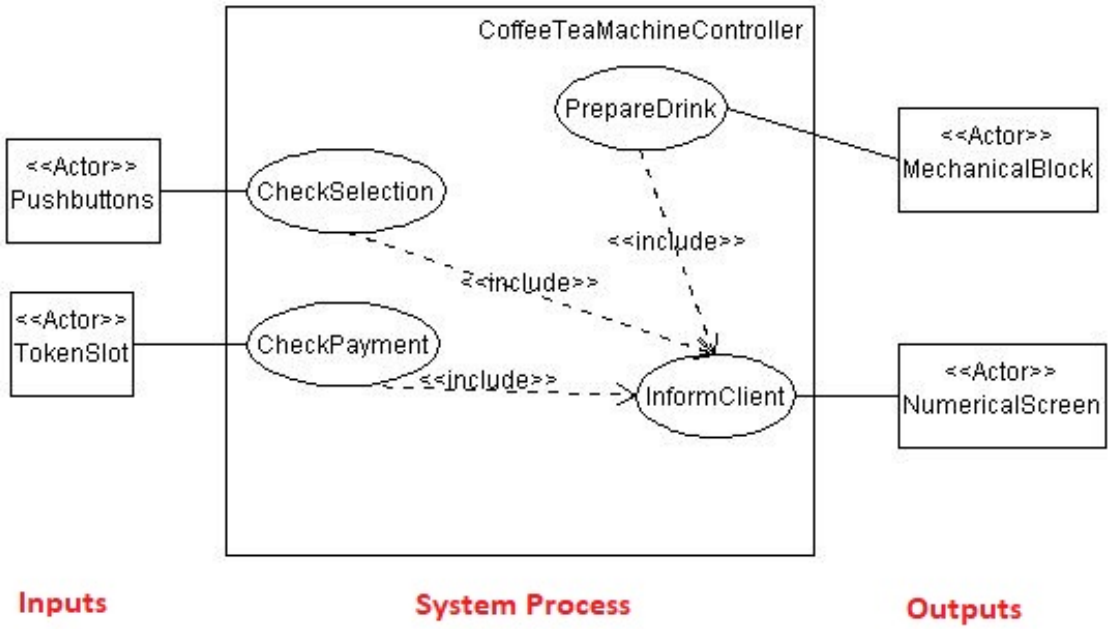

Figure 17: Use Case Diagram

9. Include: a relationship which links a function to a mandatory subfunction
10. Extend: a relationship which links a function to an optional subfunction
11. Specialization: links a ‘parent’ function to a specialized ‘child’ function

In Figure 17, on the left hand of the diagram, we have put the actors that triggers processing (i.e., inputs), while on the right we have the actors or elements that are activated as a result of it. The box in the middle represents the enclosed system: everything which is inside is what a designer promises to implement, i.e. the four bubbles are the use cases. Use cases are linked to the actor(s) involved in them.

Once the use case diagram is finished, the next step is to create an activity diagram. Just as before, right click on the diagram tab and select “New Activity Diagram”. In TTool, an activity diagram is portrayed in the form of what is commonly known as a flowchart, depicting the internal behaviour of the system. The elements that can be added to it are shown on the button tab and they include activities, partitions, the start and end of the activity, choices and tabs for whether there is a signal being sent or received.

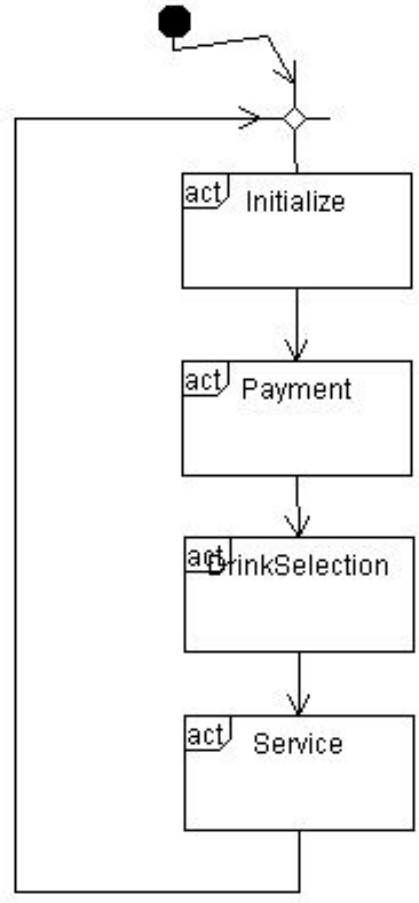

Figure 18: Activity Diagram

In the example shown in Figure 18, one start, four activities and one junction are used. After each of the elements is added, they can be edited by double clicking on them. Each activity box represent an execution stage of the system. In addition, to edit the choice

elements, click on each of the brackets or guards that appear on each of the rhombus connectors and write in the condition that must be satisfied for each option. Finally, the arrows, or associations, can be added by clicking on the respective button and then by clicking on a point on the perimeter of each of the elements being linked.

The final stage of the analysis is creating a scenario or sequence, e.g. to show how the system interacts with its environment. As before, right click on the diagram tab and select "New Sequence Diagram." This diagram will allow us to showcase the logical progression of actions as well as the different instances in which different actors or components come into play during the execution of the system and of the environment. Using the different buttons at the top of the window one can add instances, timers and indicate whether the message being transmitted is synchronous or asynchronous among other elements. To add an actor, click the "Instance" button and place it on the window. Then, double click as if to edit. When the pop up window appears, check the box that says "actor". After setting up the main structure of the system use the arrows mentioned before to represent the messages being sent between the different elements, add text describing the process by double clicking on them after setting them in place. Though not shown in this example, when adding a timer one must also indicate the amount of time assigned to it and its attributes.

As shown in Figure 19 and 20, the sequence diagram shows a visual description of the flow of messages and information inside the system. If required, one can create more than one diagram in the same window, or can add as many sequence diagrams as necessary.

5 Design

Just as with all the previous sections, to start the design, one will right click on the panel tab and select "New Design". This will open the window to create a "Block Diagram"– here we will create the main structure of our system. To add a block, click on the block button shown in Figure 21. You will create as many blocks as actors/parts you have determined in the previous parts of the creation process.

1. Edit interaction overview diagram
2. Add a comment: add a comment to the diagram
3. Security pragmas
4. Safety property
5. Performance property
6. Comment connector
7. Block
8. Crypto block
9. Add an avatar firewall
10. Data type
11. Library function

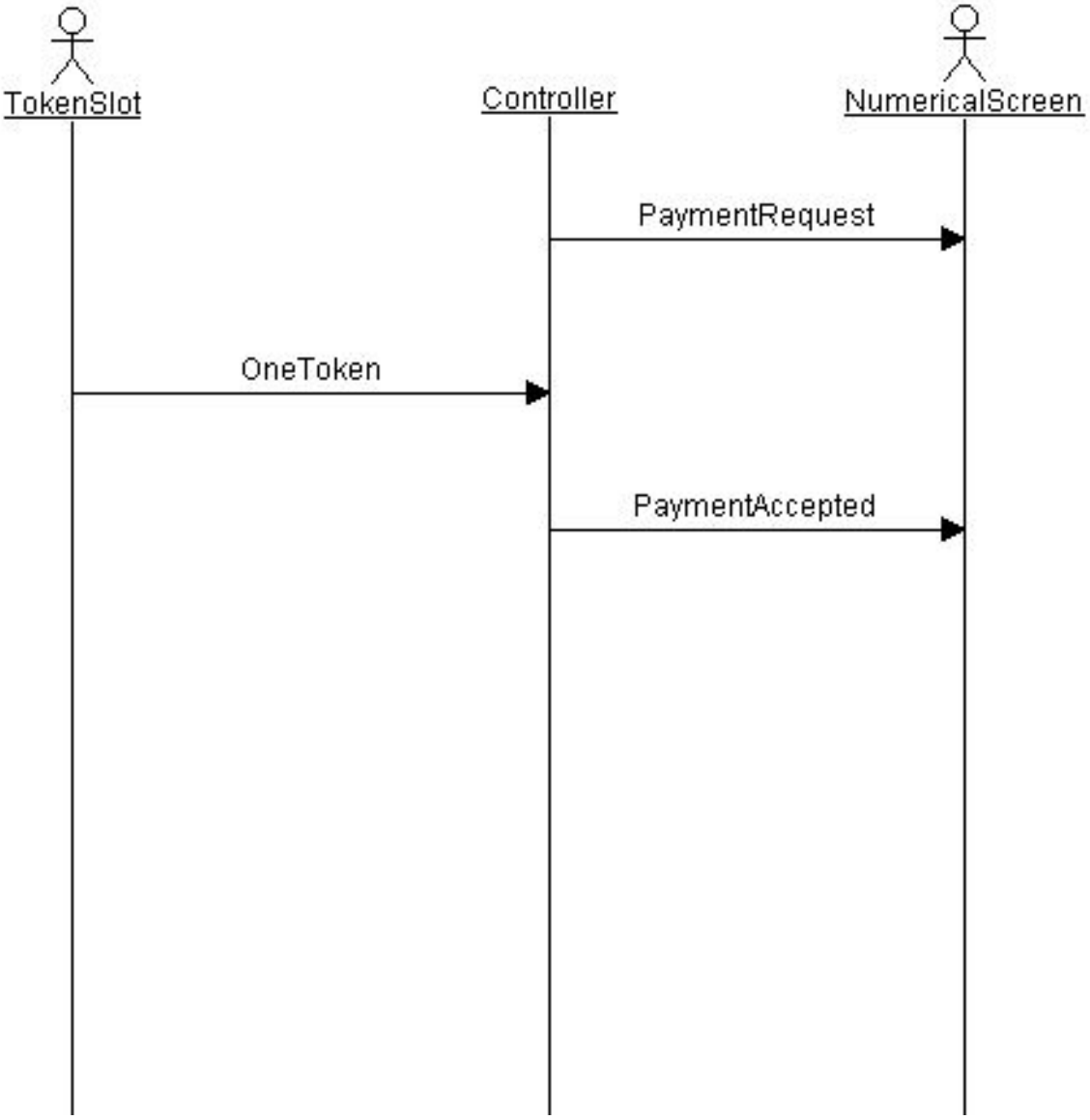

Figure 19: Logical progession of the ‘Payment’ sequence diagram

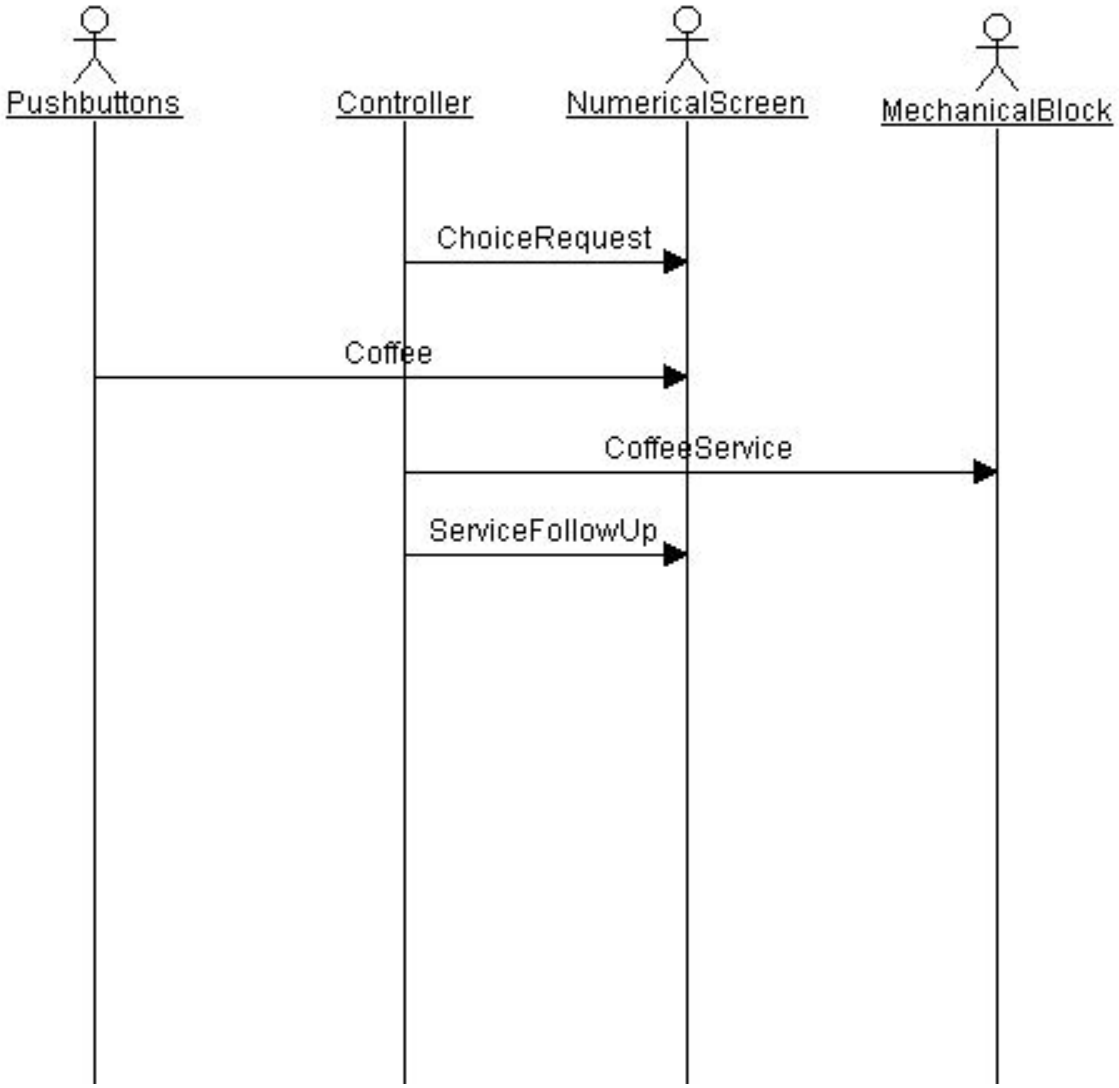

Figure 20: Logical progression of the ‘ChoiceAndService’ sequence diagram

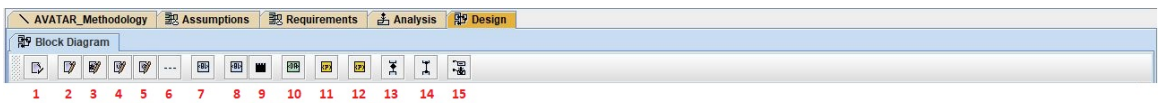

Figure 21: Structural elements of the block diagram

12. Crypto library function
13. Composition connector
14. Port connector
15. Show/hide element attributes

The coffee machine controller has five blocks: the pushbuttons, the token slot, the controller of the machine, the mechanical block, and the numerical screen. The name of each block can be edited by double clicking on the top part of the block and writing in the desired name. In addition, every time a block is created, a new tab corresponding to that block will appear next to the block diagram. We will look further into each of them later in this manual.

Following the creation of the blocks, attributes and signals for each of them will be added. To do this, double click on the bottom part of the block. A pop up window as the one shown in Figure 22 will show up. In the 'Attributes' section we will indicate the elements or variables that will interact with that section of the machine and whether they are integers, timer or Boolean values. In this example only the token slot, the controller and the pushbuttons have 'Attributes'. In the two first of them we can find "T" (short for token), which is an integer. This can be defined by selecting "int" from the drop down menu in the section indicated in Figure 22. In addition, in the two last of them we can find timers.

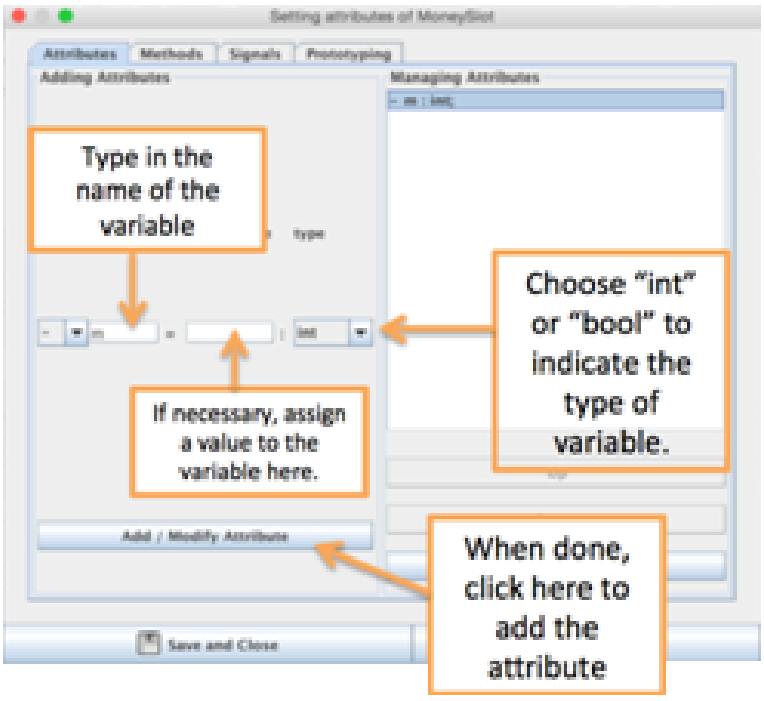

Figure 22: Setting attributes of a block

Furthermore, the 'signals' represent the information being sent and received by each component of the machine. On the same pop up window as before, click on the signals tab on the top. To add a signal, just select whether it is coming in or going out in the box shown in Figure 23 and then give it a name. If two blocks have communication between them, one should have the outgoing signal, and the other the receiving signal.

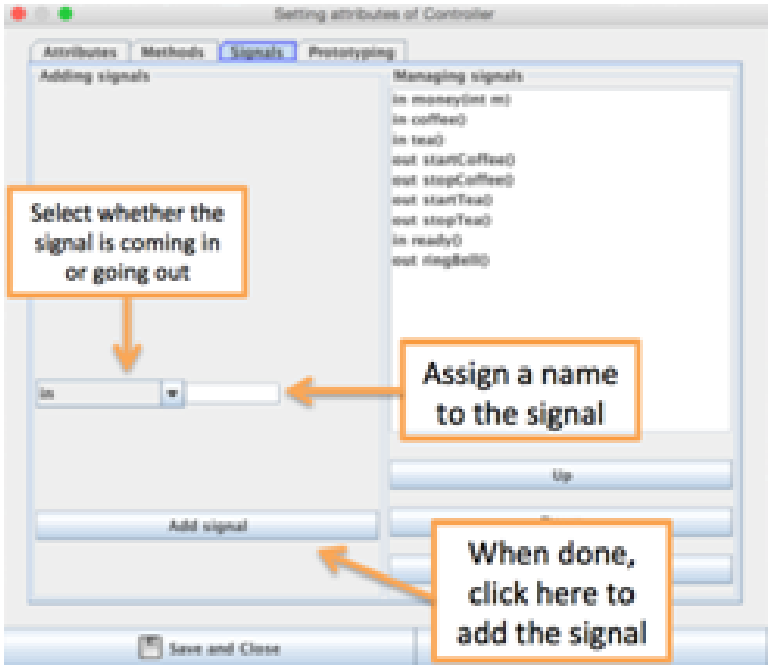

Figure 23: Setting signals of a block

Finally, one will add port connections by clicking on the respective button shown in Figure 21, and then selecting a point on the perimeter of the blocks being connected. Next, double click on the connector; a new window will pop up (Figure 24).

Here you will select the outgoing and incoming pairs of signals and one by one add them as shown in Figure 24. In addition, you can select whether this message is asynchronous or synchronous on the boxes in the inferior part of the window. Once all the signals and connections between each of the blocks are set, we can move on to the diagrams for each of the blocks. The block diagram created for the coffee machine controller is shown in Figure 25.

Once the block diagram is finished, we will go to the tabs that appeared for each of the blocks. Here, we will show through a flow chart the process through which each element goes to complete its specified task. Unlike before, in this case the black dot that indicates the process is starting is automatically added. We will complete the design from there down. Also, differently than in the activity diagram created during the analysis, here we can add the states in which the machine, or part of the machine is, choices, signals being sent and received and stops. Each of these buttons are shown in Figure 26.

1. Edit AVATAR state machine diagram
2. Add a comment: add a comment to the diagram
3. Comment connector
4. Connect two operators together
5. Start: add a starting point to the diagram
6. Stop: add a stopping point which will be the final state when it is reached (the block instance ‘dies’)

Figure 24: Port Connection between blocks

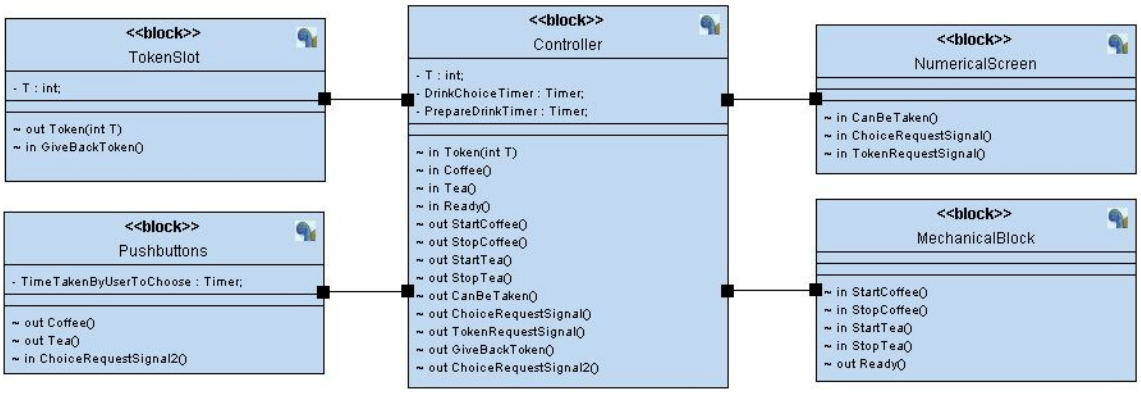

Figure 25: Block diagram

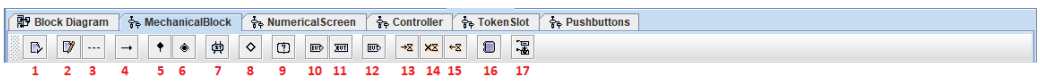

Figure 26: Structural elements of each block diagram

7. State
8. Choice: add transitions that can be taken following specific conditions
9. Select random: add an attribute that can give a random value to a variable
10. Send signal: add an output signal
11. Receive signal: add an input signal
12. Library function call
13. Set timer: add a timer that will start when it is reached
14. Reset timer: stop the timer's clock and reset it
15. Wait for timer expiration
16. Enhance
17. Show/hide AVATAR IDs

Using as reference what was already indicated in the block diagram, we will determine when messages are being sent and the states in which the machine is before and after each iteration. The diagrams for the mechanical block, numerical screen, controller, token slot and pushbuttons can be observed as example in Figures 27 to 31 respectively.

6 Verification

Finally, the last step of the process is to verify that the system we have created works properly and follows each of the steps we desire it to. In order to do this, we will use the "Syntax Analysis" tool, which is shown in Figure 32. After you click on it, a screen like the one in this same figure will appear. There you can select whether you want to check the syntax of the whole system or individual parts. After this is determined, click on start syntax analysis and wait a second. After the software is done, it will either pop up an error message like the one in Figure 33, meaning that the system has not been designed correctly; or if no errors are found, the "Interactive Simulation" button (Figure 32), which was once greyed out, will now be blue and you will be able to click on it.

1. Syntax Analysis
2. Select elements to analyze
3. Start analysis
4. Interactive Simulation (currently greyed out)

If this is the case, then you will click on it and a window like the one in Figure 34 will appear on your screen. Click on run simulation and wait. The software will then go through a full iteration of the machine and show each step in the way portrayed by Figures 35 and 36 below. This tool is very useful in that it allows us to see how the different parts of the machine communicate as well as each step of the process programmed.

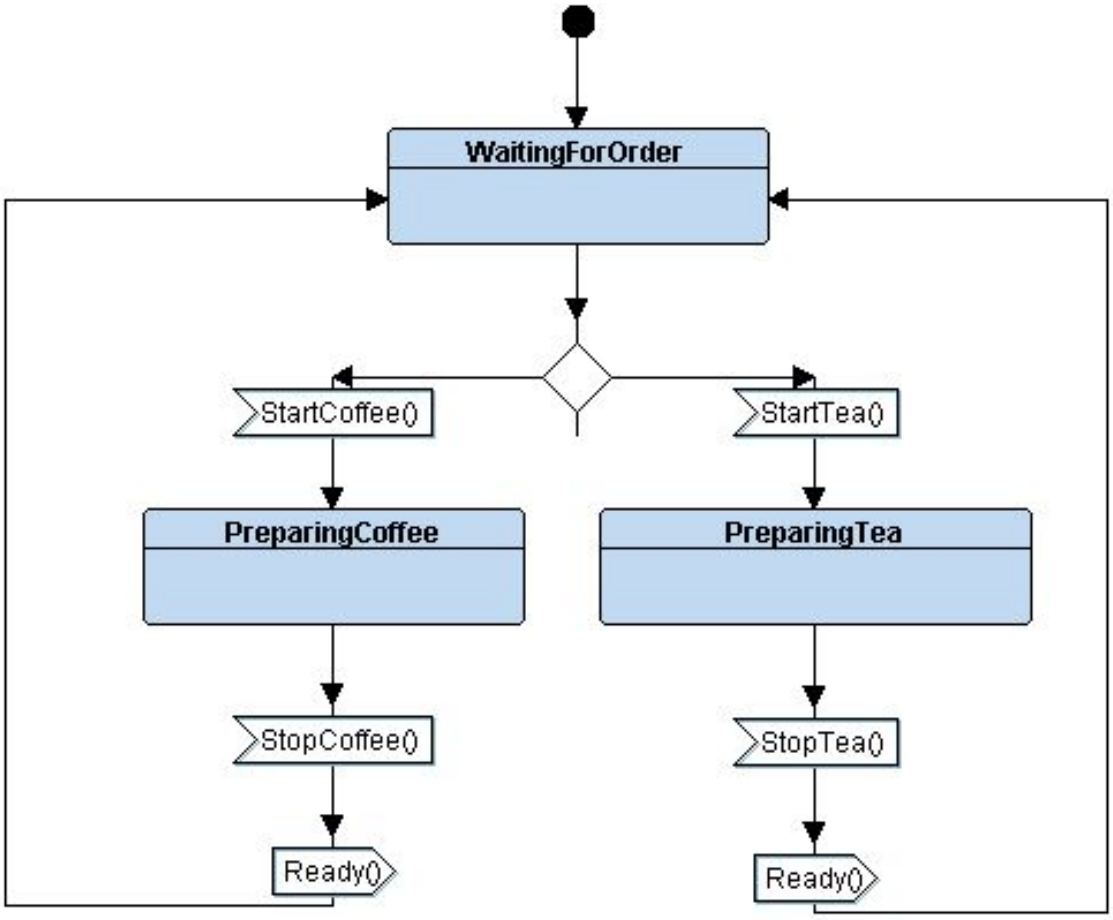

Figure 27: Mechanical block diagram

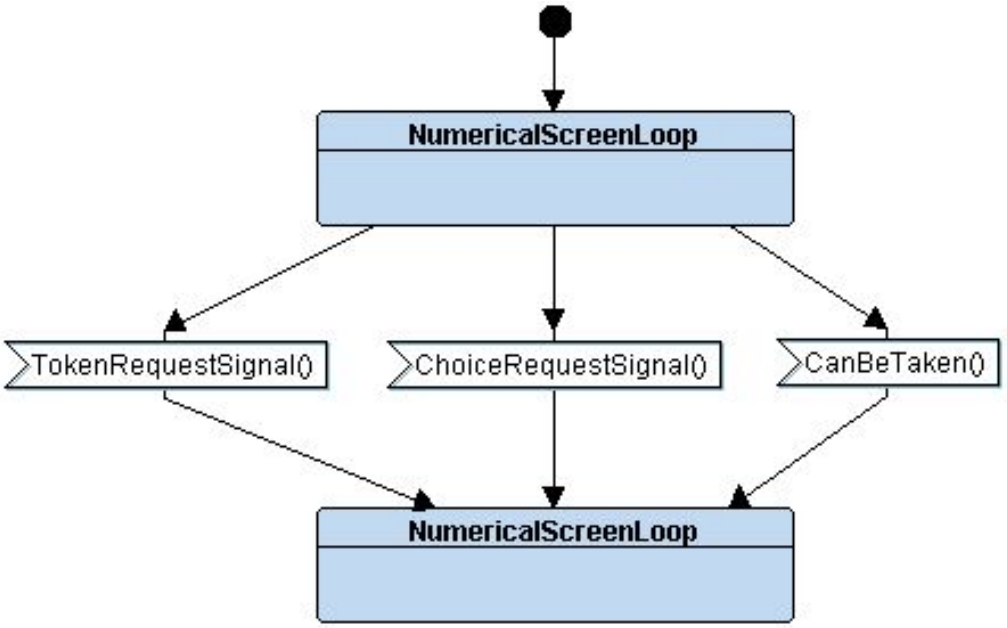

Figure 28: Numerical screen diagram

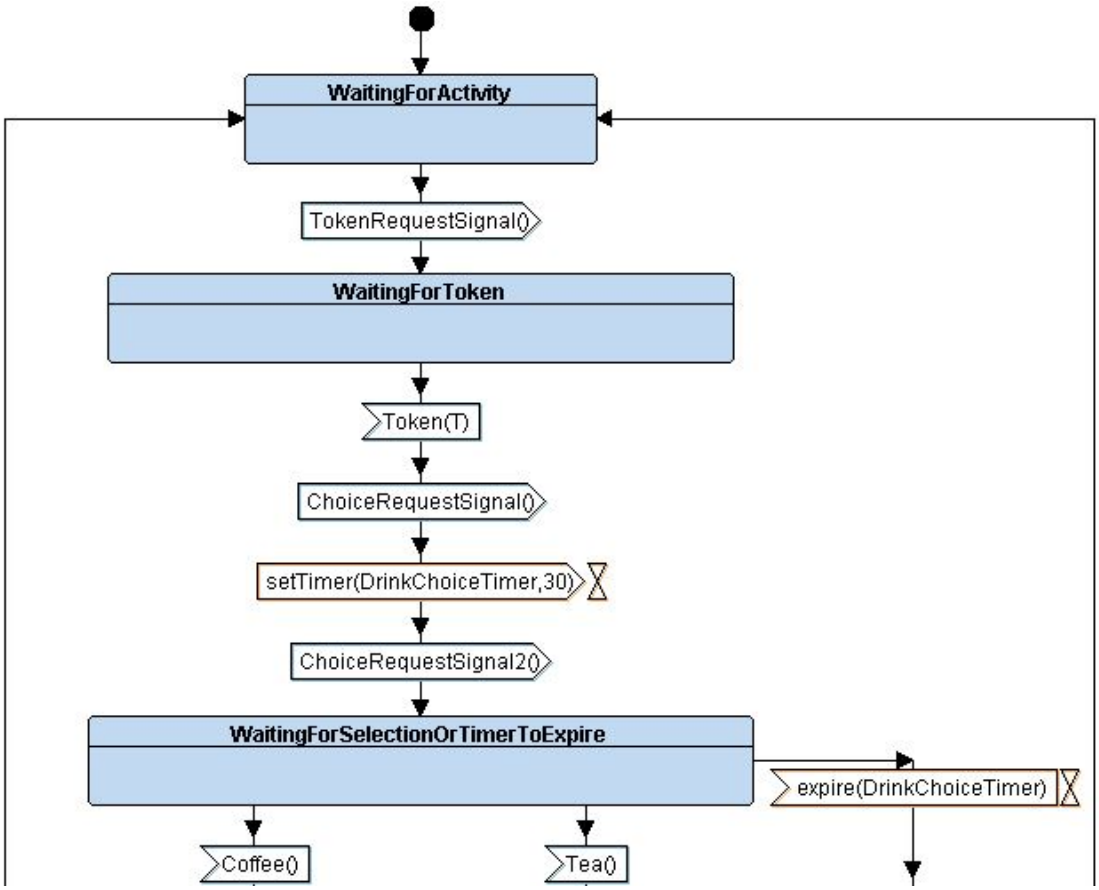

Figure 29: Controller diagram

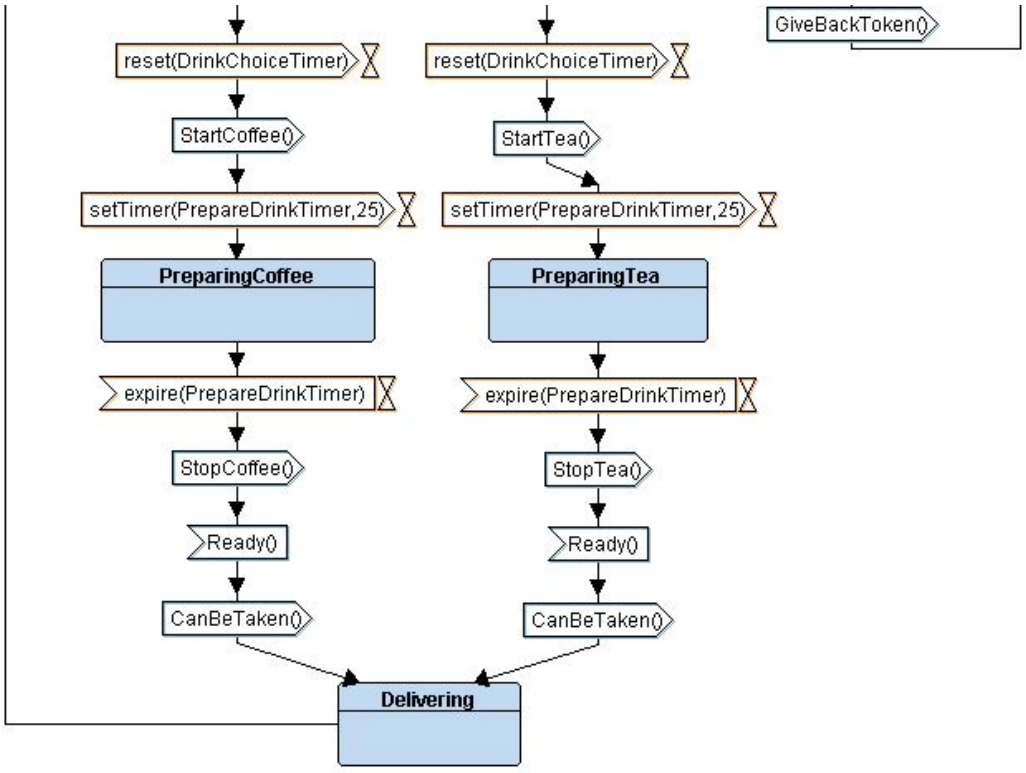

Figure 30: Token slot diagram

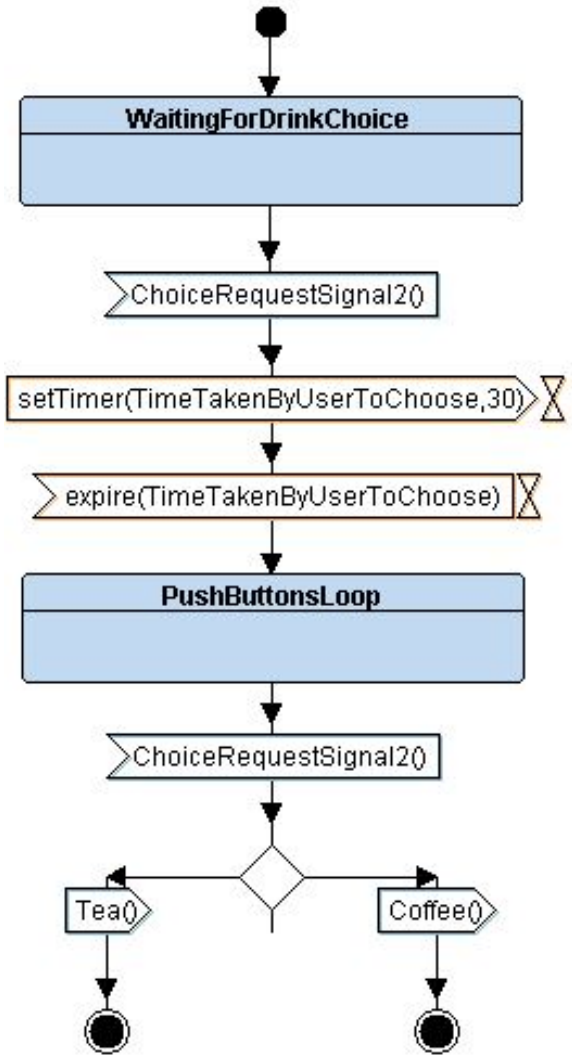

Figure 31: Pushbuttons diagram

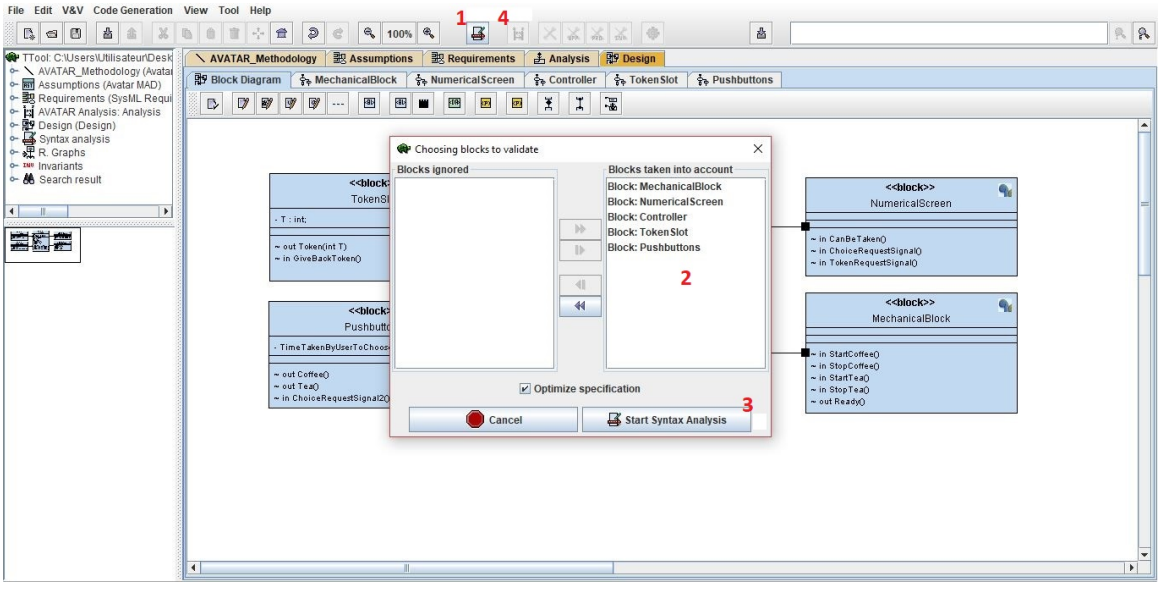

Figure 32: Syntax analysis

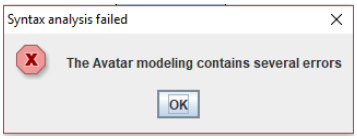

Figure 33: Syntax analysis error message

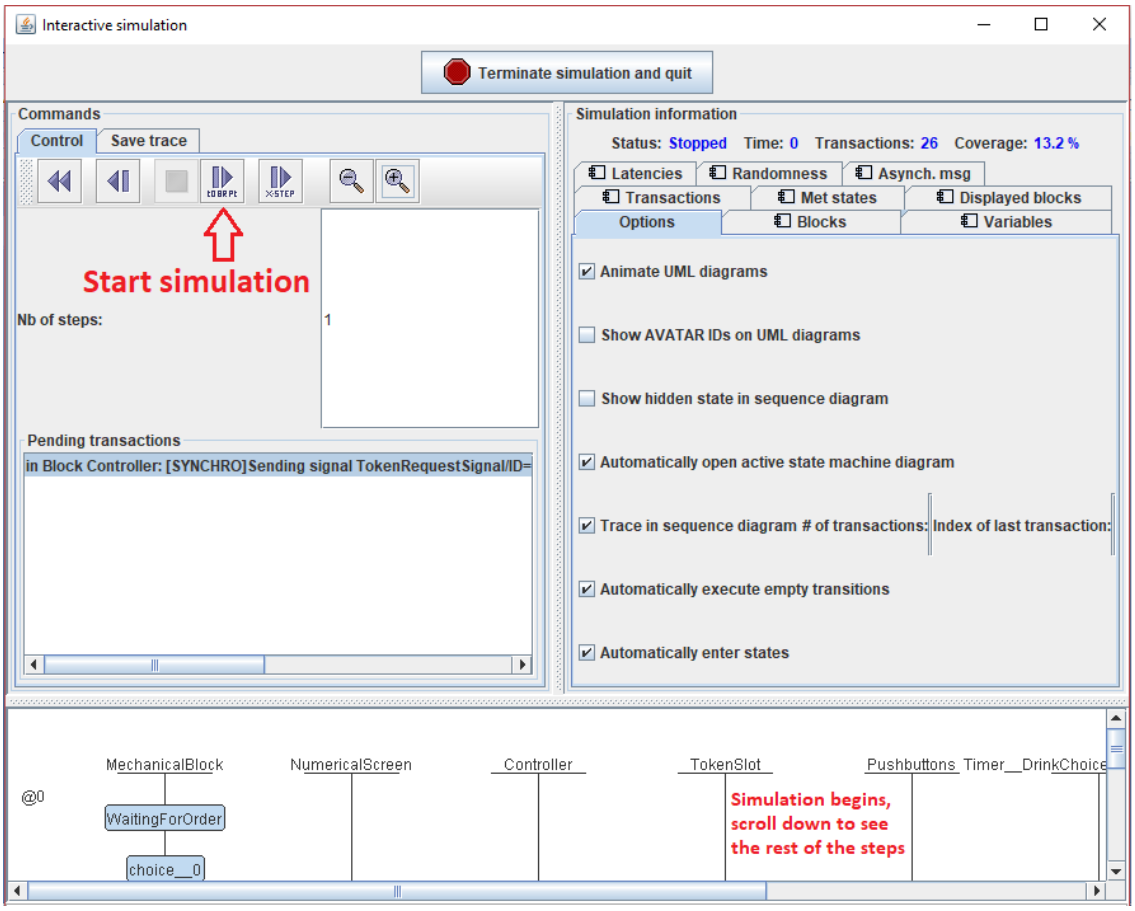

Figure 34: Interactive Simulation

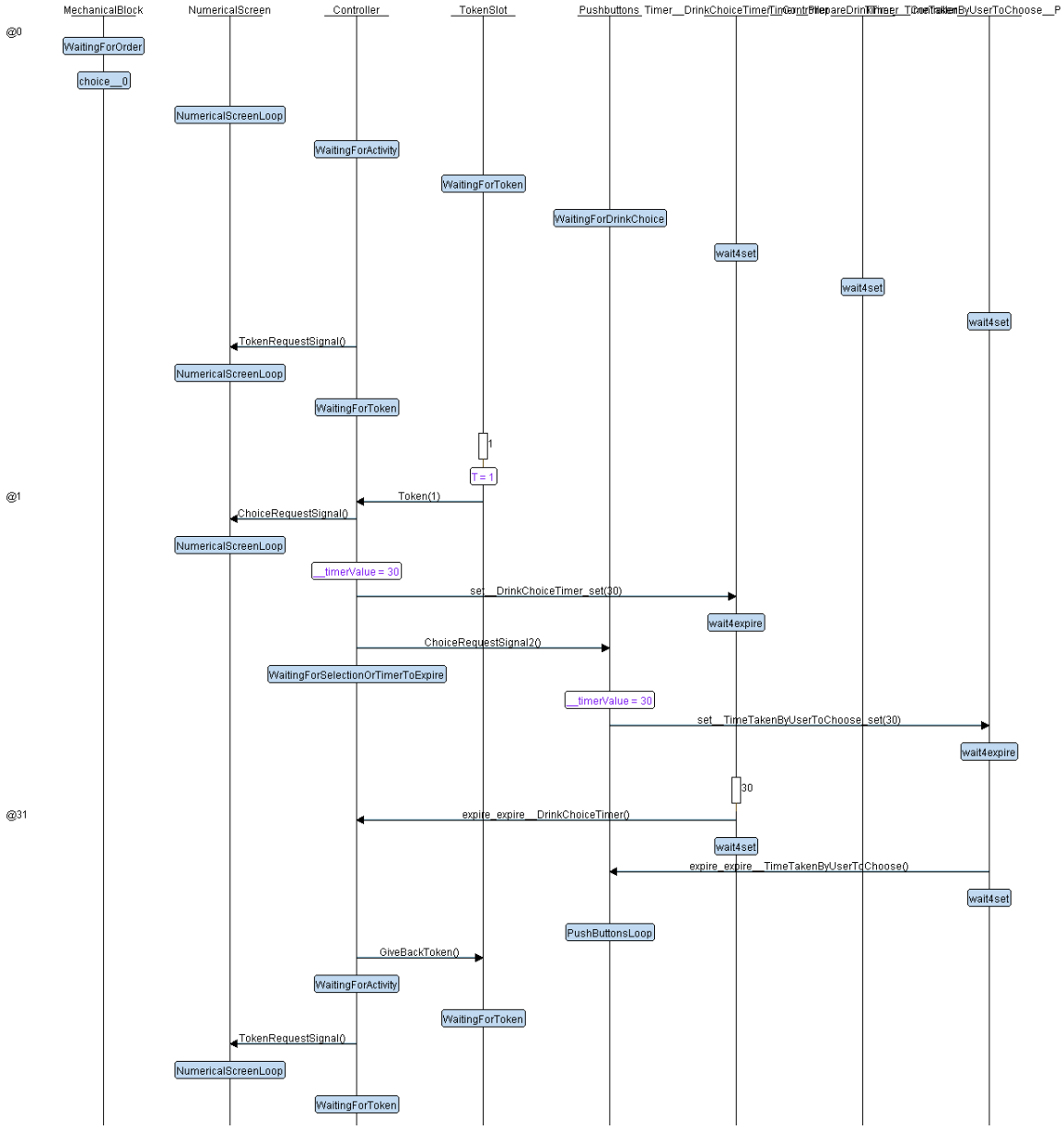

Figure 35: Simulation steps (part 1)

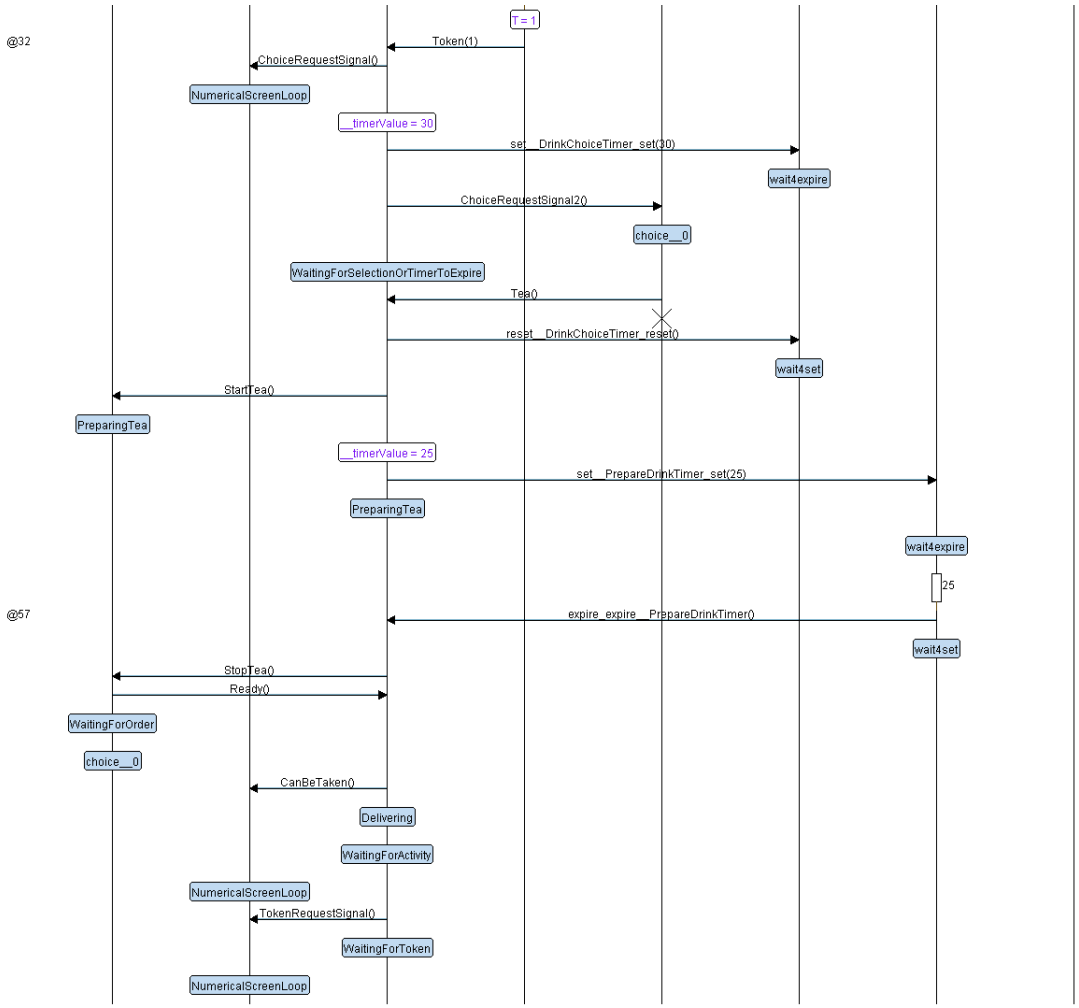

Figure 36: Simulation steps (part 2)

6.1 Reachability Graphs

Another interesting feature in the TTool software is the ability to graph or map the processes. This generates a graph that allows us to see all the different pathways that the system follows to reach its final state, each showing a different scenario. In addition, one can select a specific initial and final state, which simplifies said graph and gives a neater depiction of the process. Figure 37 shows an example.

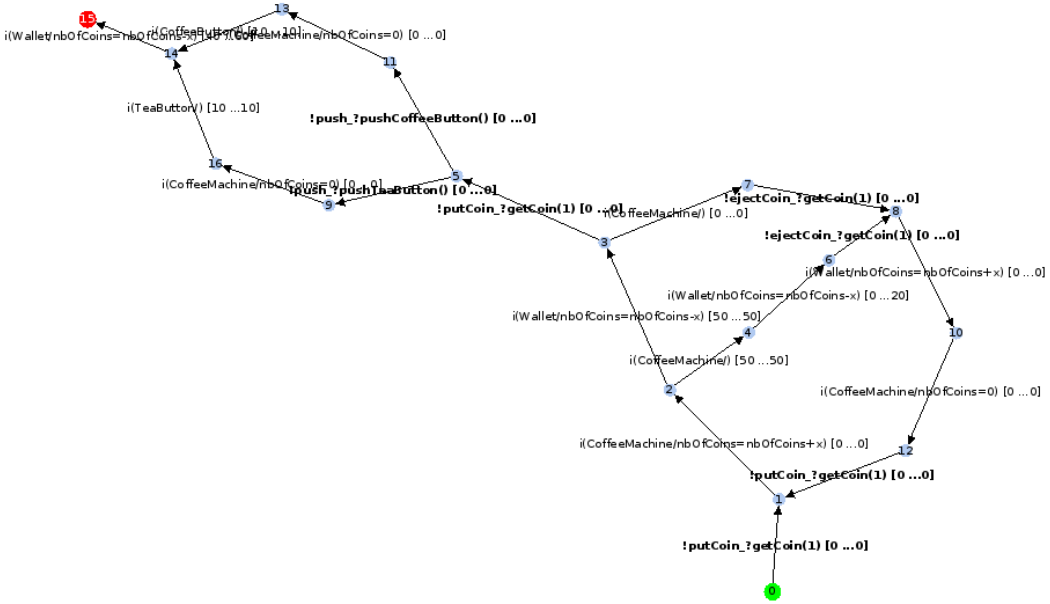

Figure 37: Reachability graph

6.2 Safety pragmas

Safety pragmas can be inserted in the model. These pragmas follow the following grammar.

⟨*pragma*⟩ ::= ⟨*path*⟩ ⟨*state*⟩ ⟨*property*⟩
| ⟨*property*⟩ ⟨*leadsto*⟩ ⟨*property*⟩

⟨*path*⟩ ::= ‘A’ | ‘E’

⟨*state*⟩ ::= ‘<>’ | ‘[]’

⟨*leadsto*⟩ ::= ‘->’

⟨*property*⟩ ::= ⟨*stateproperty*⟩
| ⟨*intproperty*⟩
| ⟨*boolproperty*⟩
| ⟨*property*⟩ ⟨*binaryop*⟩ ⟨*property*⟩

⟨*stateproperty*⟩ ::= ‘BlockName.stateName’

⟨*intproperty*⟩ ::= ⟨*intexpr*⟩ ⟨*intcomparator*⟩ ⟨*intexpr*⟩

⟨*boolproperty*⟩ ::= ⟨*boolexpr*⟩ ⟨*boolcomparator*⟩ ⟨*boolexpr*⟩
| ⟨*boolexpr*⟩

⟨*intexpr*⟩ ::= ‘BlockName.integerattribute’
| ⟨*intvalue*⟩

⟨*intexpr*⟩ ::= ‘BlockName.booleanattribute’
| ⟨*booleanvalue*⟩

⟨*binaryop*⟩ ::= ‘&&’ | ‘||’

⟨*intvalue*⟩ :: ⟨*integer*⟩

⟨*booleanvalue*⟩ :: ‘true’ | ‘false’

⟨*intcomparator*⟩ ::= ‘<’ | ‘>’ | ‘==’ | ‘!=’

⟨*boolcomparator*⟩ ::= ‘==’ | ‘!=’

6.3 Latency Analysis

Many real-time safety-critical systems interact continuously with the environment and users. New input from the outside world is processed by the system, which then effectuates a response observable in the real world. The timing of such responses can greatly impact functionality and safety.

Latency analysis is performed in multiple steps. First, the quantitative requirement on latency must be defined. Next, the text requirement must be translated to determine which modeling elements the critical events refer to, after which latencies can be measured in simulation. The results are then conveniently displayed on the modeling diagrams. This example uses the updated Coffee Machine model that can be found in the TTool sample modeling repository.

Figure 38 shows a requirement related to timing: The Coffee Machine must take less than X seconds to finish a drink command after payment has been received.

The operators relating to this requirement are the last ‘getCoin(x)’ and the idle state after processing ‘WaitingForFirstCoin’. Figure 39 shows how the two operators are tagged with ‘Latency Checkpoints’, in the form of blue flags.

After syntax analysis, run interactive simulation, and then on the Latency Panel (Figure 40), we indicate that we should measure the latency between these two checkpoints by selecting the operators and clicking the ‘Add latency’ button. After running 100 steps of the

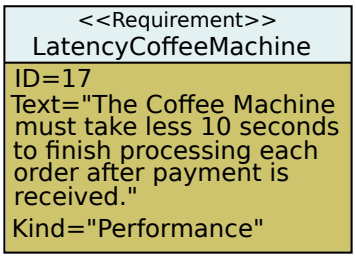

Figure 38: Latency Requirement for Coffee Machine

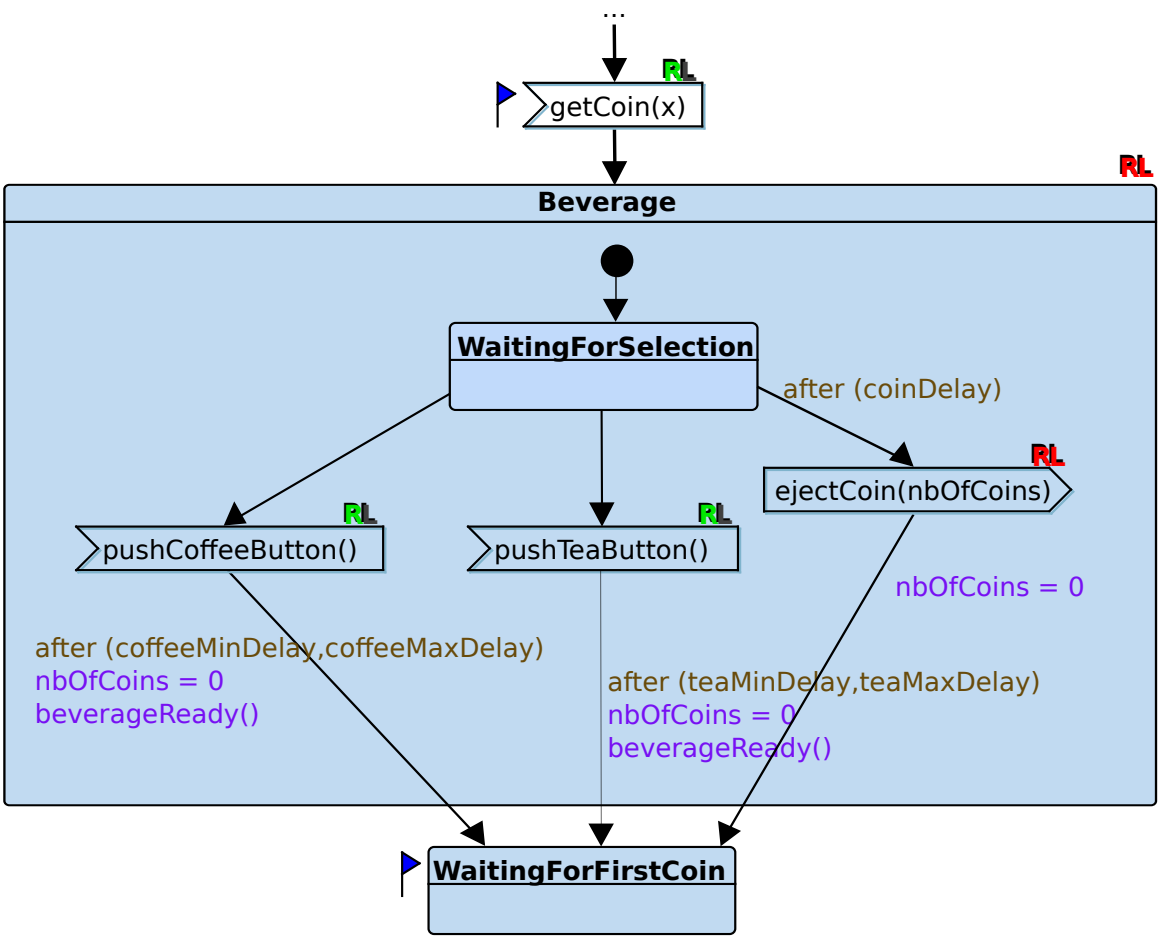

Figure 39: Latency Checkpoints in State Machine Diagrams

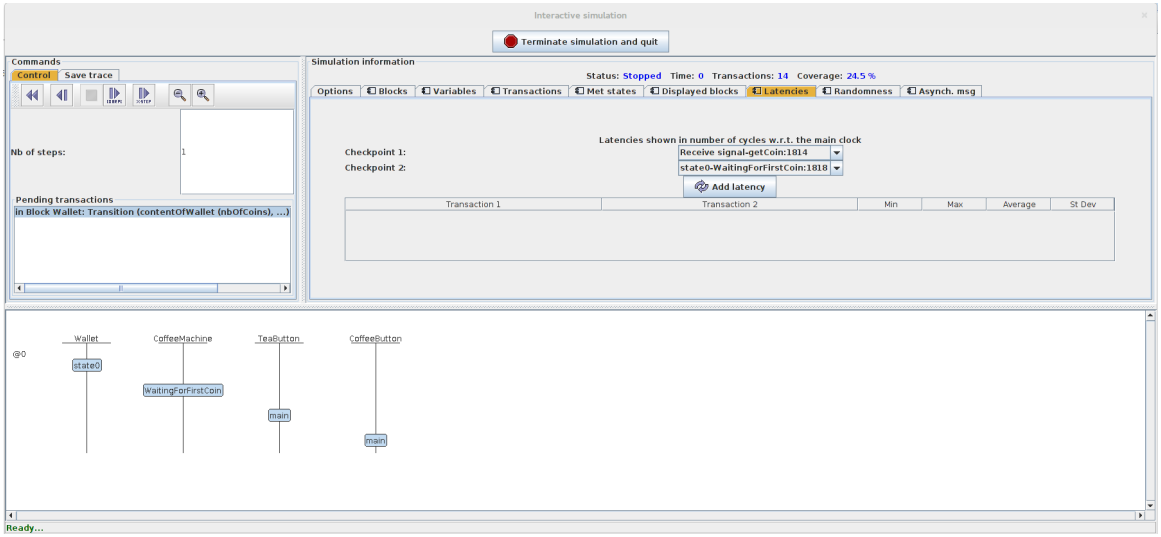

Figure 40: Latency Measurement Panel

simulation, the minimum, maximum, average, standard deviation of the latency measurements is displayed as shown in Figure 41.

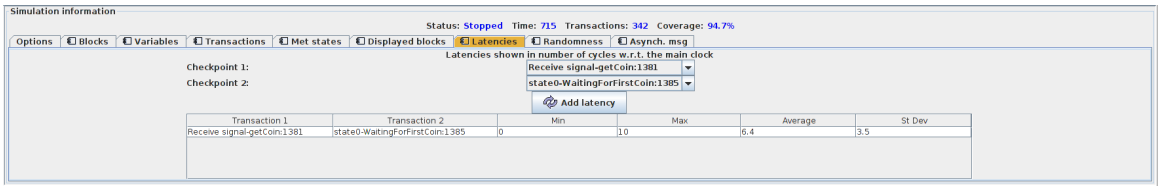

Figure 41: Latency Results

7 Conclusion

All in all, TTool is a useful software, which allows the user to generate systems using SysML. By guiding the user through each of the steps of the process (Methodology, Assumptions, Requirements, Analysis and Design) it serves as a platform for better structured, and comprehensive designs. These models provide all the information required to understand the present state of the design and enable easy changes in the future. It is straightforward and intuitive and it has many more features than the ones shown in this manual, therefore making it a very complete tool in the development and simulation of engineering designs.