# **Stimulus** Tutorial

## ***Porte Automatique***

Jean-Michel Bruel, bruel@irit.fr

Version 1.0, 2018-11-07

# Table des Matières

Dans ce document vous trouverez:

- son contexte en Section 1,
- une étude de cas commençant en Section 2,
- des explications sur les éléments de base de Stimulus en Section 7 (uniquement en anglais pour l'instant), et
- des explications sur les éléments avancés de Stimulus en Section 8 (uniquement en anglais pour l'instant).

Une **F**oire **A**ux **Q**uestions (FAQ) est disponible en Section 9 (uniquement en anglais pour l'instant).

# 1. Contexte

## 1.1. À propos de ce tutoriel

Ce tutoriel est basé sur le tutoriel officiel (en anglais) d'Argosim et disponible ici : https://download.argosim.com/index.php/s/5ZszF09tl0rd4gv/download.

Il va permettre d'aborder :

- des aspects méthodologiques et organisationnels ;
- comment créer et utiliser des glossaires ;
- comment écrire des exigences complexes ;
- comment les debugger ;
- comment écrire des scénarios pour générer des *inputs* ;
- comment transformer les exigences en *Observers*.

## 1.2. Installation et contraintes

Ce tutoriel utilise la version `2018.09.1` de Stimulus.

1. Lancer l'installeur Stimulus
2. Générer le nombre requis pour obtenir la licence
3. L'envoyer par email pour obtenir le fichier de licence

L'outil nécessite Windows! Et les machines virtuelles ne sont pas autorisées.

# 2. Étude de cas

## 2.1. Description

Nous allons modéliser une porte automatique (type supermarché) dont le comportement peut être modélisé par le chronogramme illustré en Figure 1. Le capteur (`sensor`), indique selon que quelqu'un (`someone`) ou non (`nobody`) est détecté et si la porte doit alors être ouverte (`open`) ou fermée (`closed`).

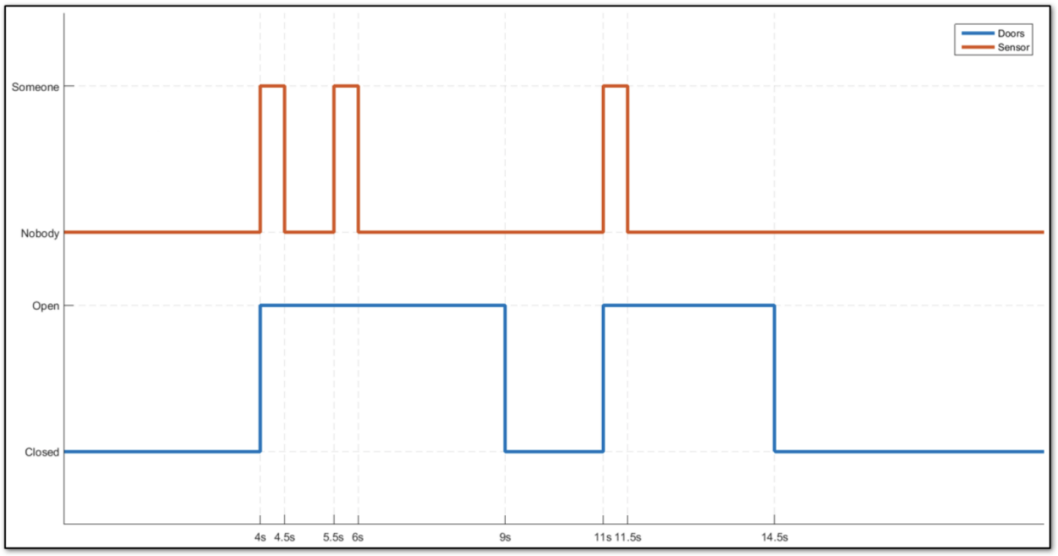


*Figure 1. Automatic door chronogram*

## 2.2. Structuration du projet

Créez les *packages* suivants:

- "Library" pour placer ses propres définitions et *templates*.
- "Requirements" pour organiser les exigences.
- "Test" pour définir les scenarios et tests.

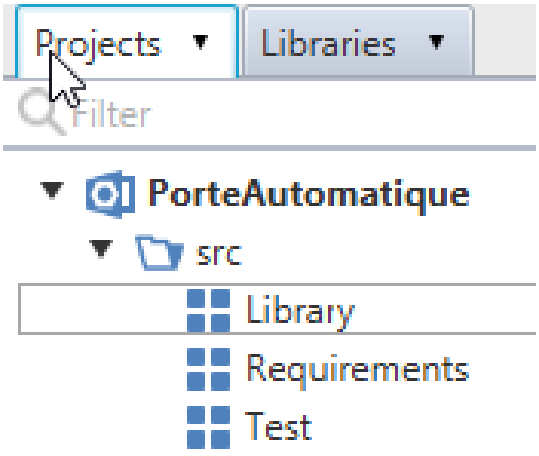


*Figure 2. Project structure*

## 2.3. Un premier Glossaire

Créez un nouveau Glossaire dans le *package* `Requirements`:

- Cliquez pour selectionner le *package* `Requirements`.
- Cliquez sur le menu **File › New › Glossary**.

Déterminez les 2 signaux (et leurs valeurs) à partir de la description Section 2.1.

Pour ajouter ces 2 definitions, cliquez sur le bouton `+`. Choisir le type `Enum`. Double cliquer pour éditer `EnumItem` et tapez la 1ère valeur possible. Ajoutez les suivantes (une seule en l'occurence dans notre exemple) en cliquant sur le bouton `+`.

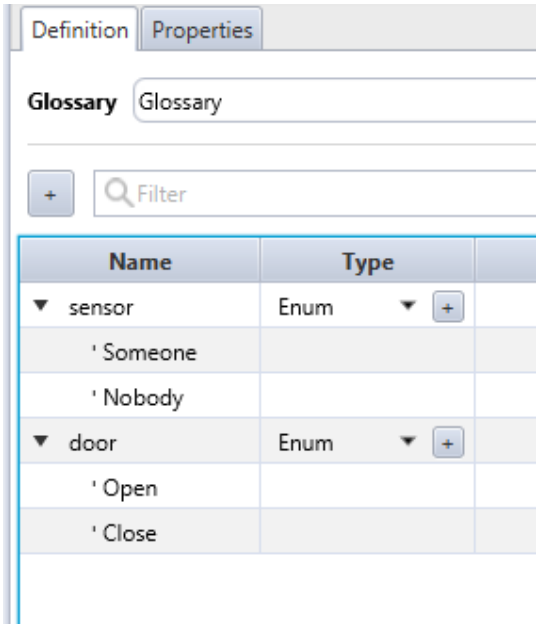


*Figure 3. Glossary*

# 3. Exigences

## 3.1. Utilisation du Glossaire

Pour créer notre 1ère exigence :

- Cliquez pour sélectionnerle *package* `Requirements`
- Cliquez sur le menu **File › New › Requirement**.
- Nommez-le "Req001".

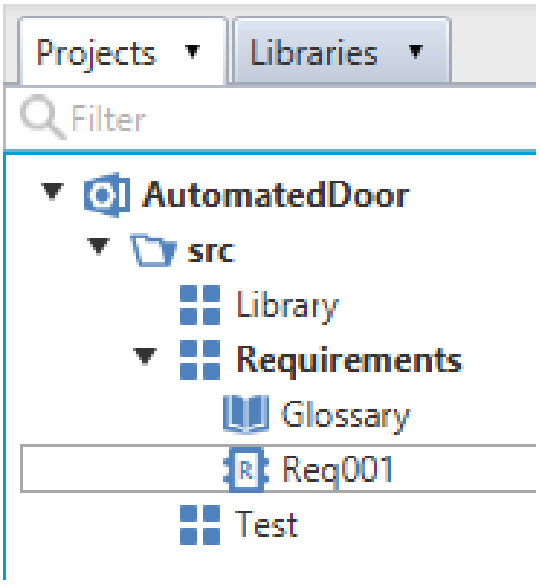


*Figure 4. First requirement*

Pour utiliser le glossaire, faire un *drag and drop* depuis l'arbre de navigation du projet vers l'interface du système comme illustré en Figure 5.

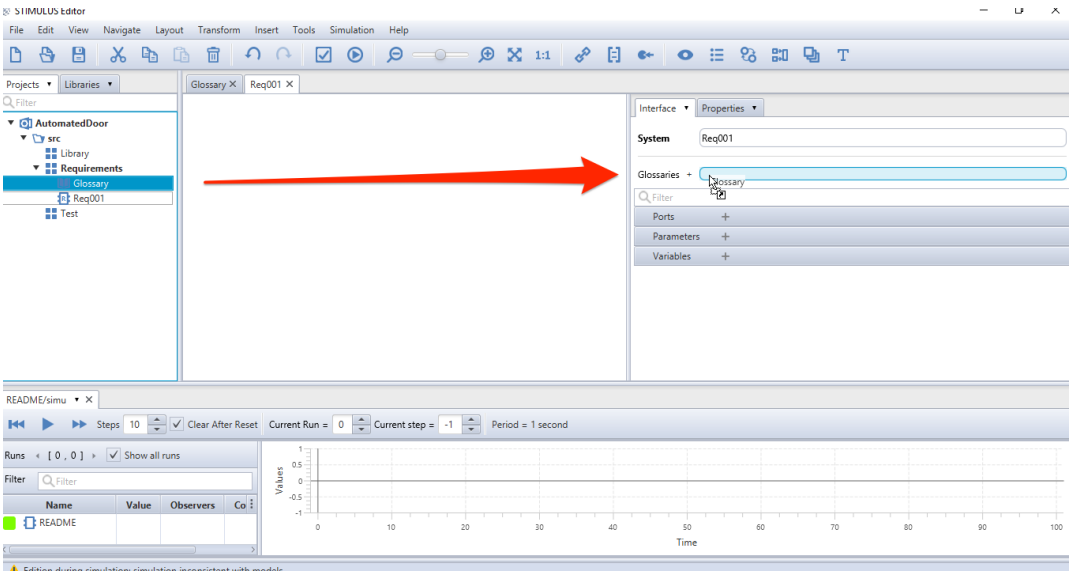


*Figure 5. Use a Glossary*

Le bouton `Glossary chooser` permet de déclarer les ports à partir des définition du glossaire (cf. Figure 6).

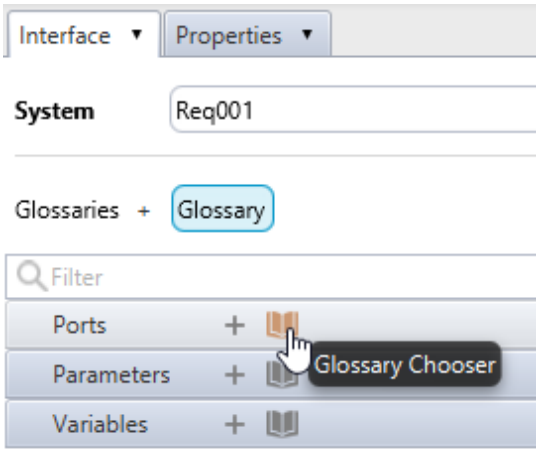


*Figure 6. Open glossary chooser*

## 3.2. Définition de l'exigence

En utilisant les éléments **Temporal › When** et **Logical › Equal** de la librairie standard, expimez les exigences suivantes : "When there is somebody, door is open" et "When there is nobody, door is closed".

Vous obtiendrez alors l'exigebce illustrée en Figure 7.

Sur la figure, le format des exigences a été modifié (click droit **Formats**)...

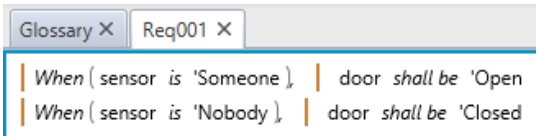


*Figure 7. Completed requirement*

## 3.3. Simulation 1

Exécutez une simulation commme illustré en Figure 8.

Observez que le comportement ne reflète pas le delai, attendu après que la porte soit ouverte `'Open`.

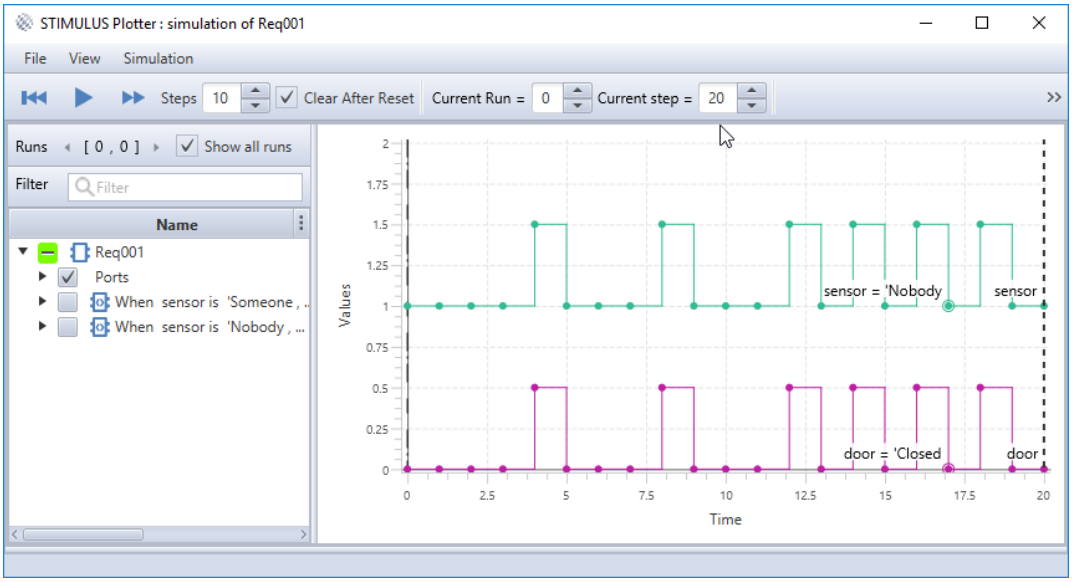


*Figure 8. First door simulation*

# 3.4. Re-définition de l'exigence

En utilisant la fonction **Temporal › ForPeriod** et le *drag & drop*, ajoutez un délai de 3 secondes comme illustré en Figure 9.

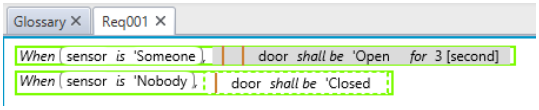


*Figure 9. Completed requirement*

# 3.5. Simulation 2

Exécutez une simulation commme illustré en Figure 10.

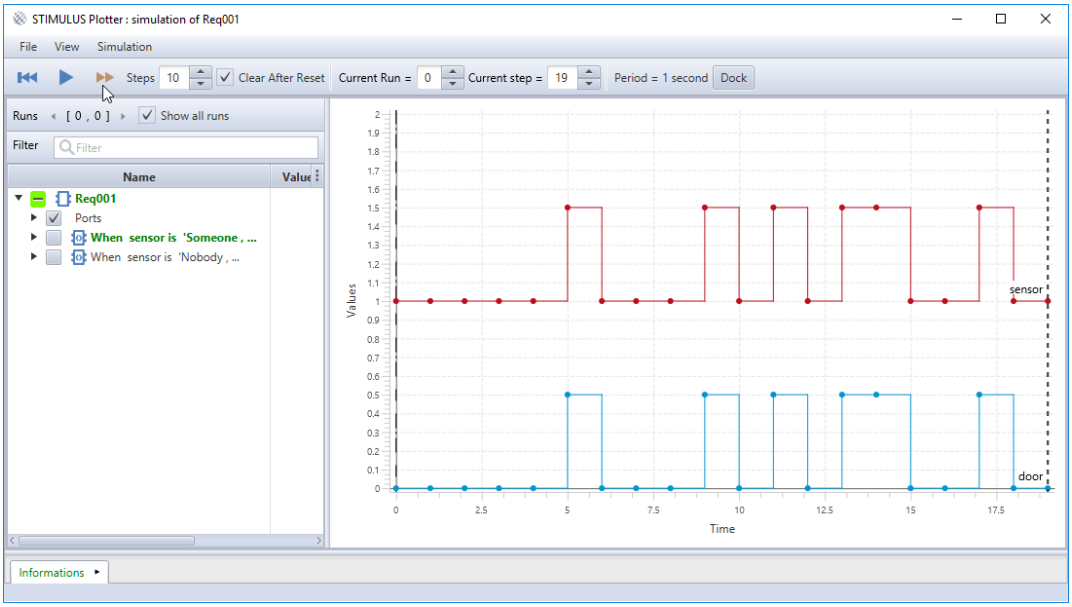


*Figure 10. Second door simulation*

Observez que le comportement ne reflète toujours pas le delai! Pourquoi ? Pour le comprendre, sélectionnez dans la simulation le pas step où `door` change de `'Open` à `'Closed`. Dans la partie `Req001`, on observe que la phrase `For 3 [second], Door shall be 'Open` n'est pas active à ce pas de simulation, comme illustré en Figure 11. C'est à cause du `When` qui désactive son `<BODY>` quand la condition est fausse!

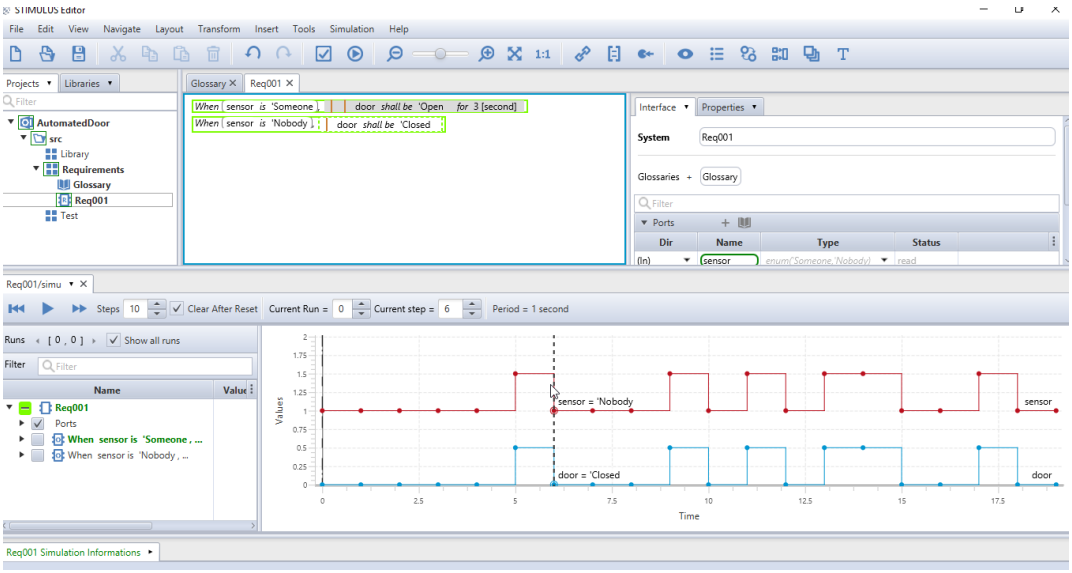


*Figure 11. Highlighted requirements*

# 3.6. Correction de l'exigence

En fait nous aurions dû écrire : "From the time someone is detected, then we shall do something for 3 seconds". Faite un *drag & drop* de la fonction **Temporal › From** sur la première exigence `When`pour la remplacer et obtenir le résultat illsuté en Figure 12.

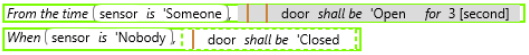


*Figure 12. Replace "When" by "From"*

# 3.7. Simulation 3

Exécutez une simulation commme illustré en Figure 13.

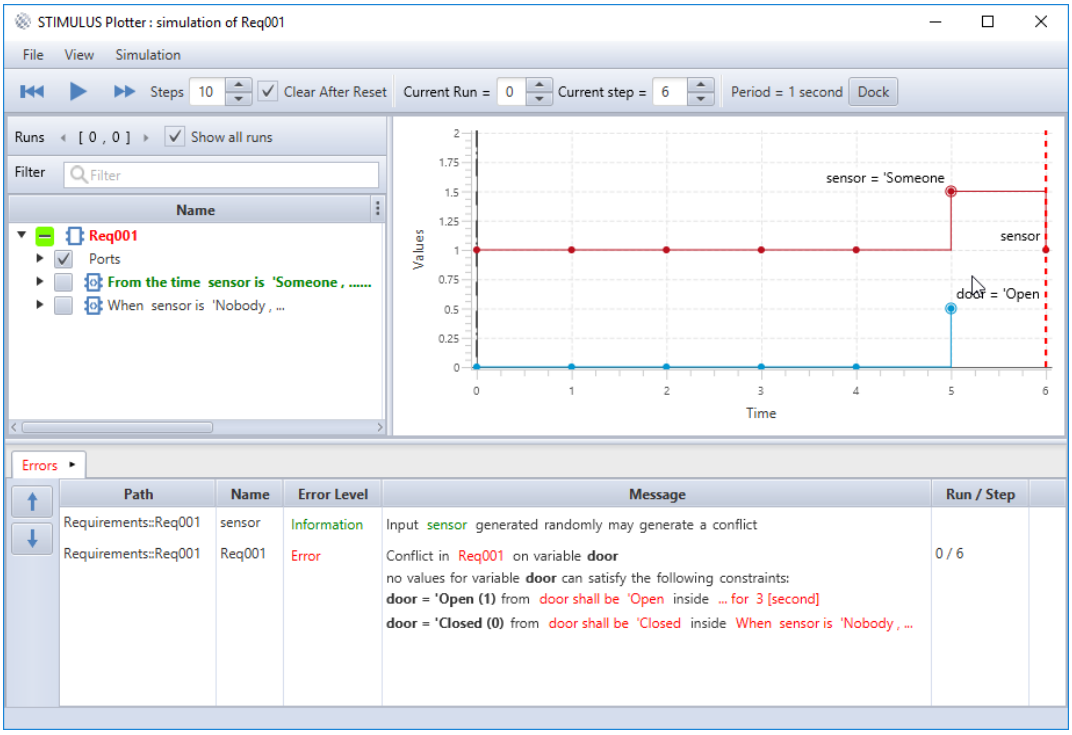


*Figure 13. Third door simulation*

Stimulus détecte un conflit en Figure 13!! Trouvez et corrigez l'erreur.

## 3.8. Nouvelle correction de l'exigence

Tentons de supprimer le deuxième terme de l'exigence, devenu obsolète (le sélectionner et cliquer sur **[ Delete ]** sur le clavier.

## 3.9. Simulation 4

Exécutez une simulation commme illustré en Figure 14... Que constatez-vous ?!

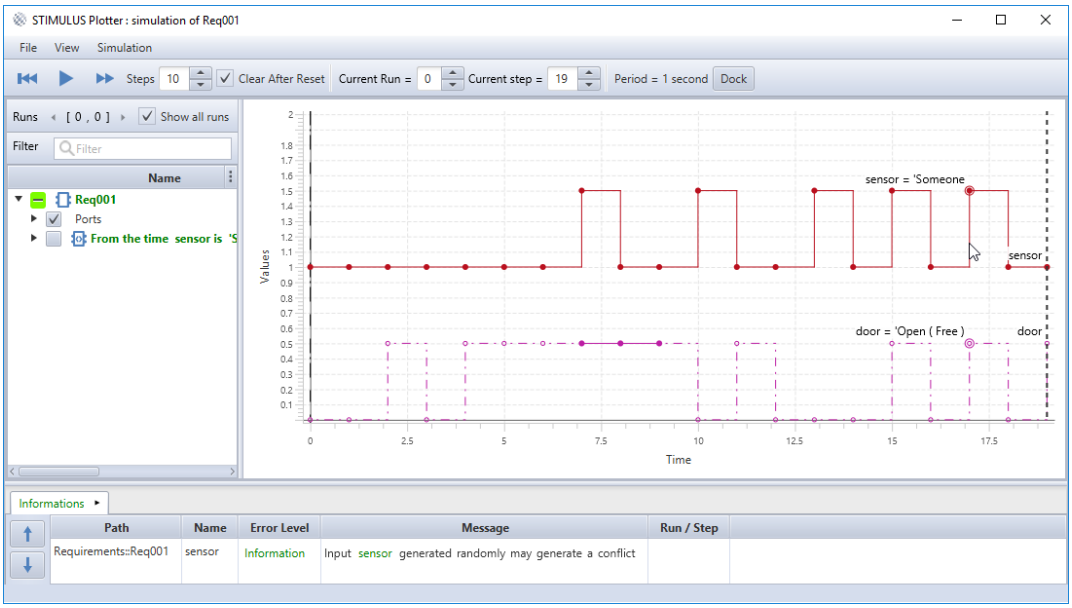


*Figure 14. Fourth door requirements*

# 3.10. Simulation 5

On observe des comportements non désirés (porte qui reste ouverte plus que 3s, qui ne s'ouvre pas après une deuxième détection, etc.). La ligne en pointillée indique qu'à ces moments, la valeur de `Door` (Open ou Closed), n'est pas contraintes. L'erreur vient du fait qu'on voulait en fait répéter le comportement "open for 3 seconds" à chaque détection de nouvelle personne!

Sélectionnez l'item **Temporal › FromEach** dans la librairie et lachez-le sur `From`. Puis relancer la simulation.

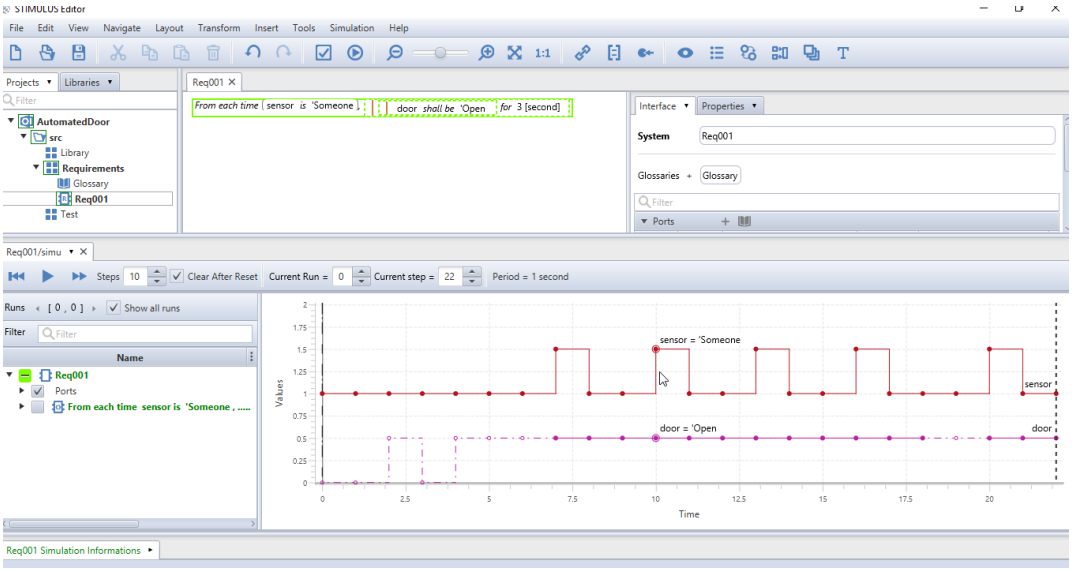


*Figure 15. Fifth simulation*

# 3.11. Simulation 6

Bon, la porte reste enfin ouverte 3s à chaque détection. Néanmoins elle reste parfois ouverte plus longtemps. Ceci peut être exprimé en utilisant l'item **Temporal › DoAfterwards**. Utilisez-là pour obtenir le résultat illustré en Figure 16 et qui donne la simulation illustrée en Figure 17:


*Figure 16. Sixth door requirements*

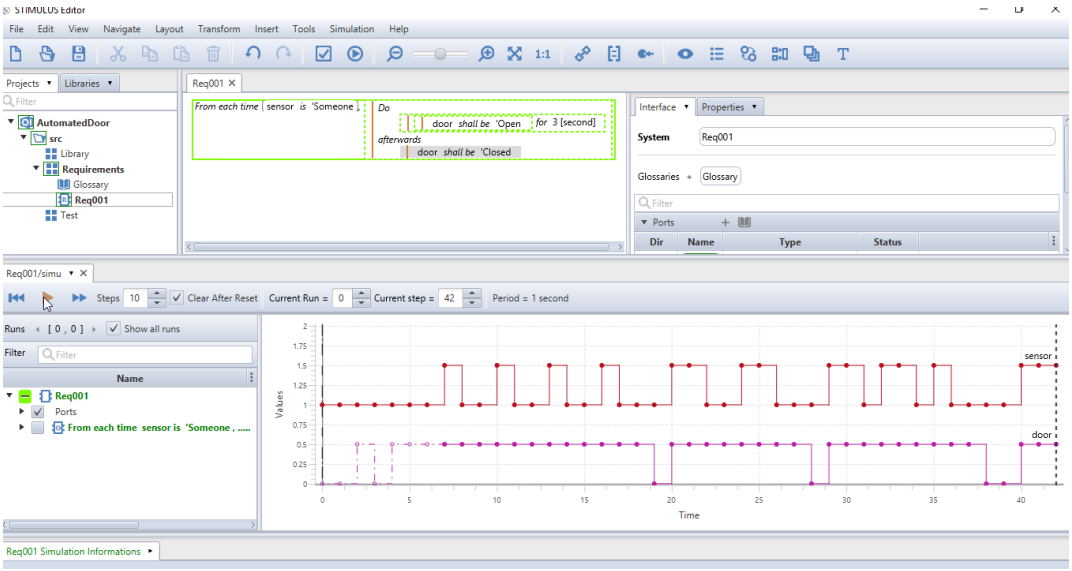


*Figure 17. Sixth door simulation*

## 3.12. Simulation 7

Ajoutez l'exigence manquante qui permette d'éviter l'ouverture inutile de la porte en début de simulation. Vous devrez obtenir une simulation similaire à la Figure 18.

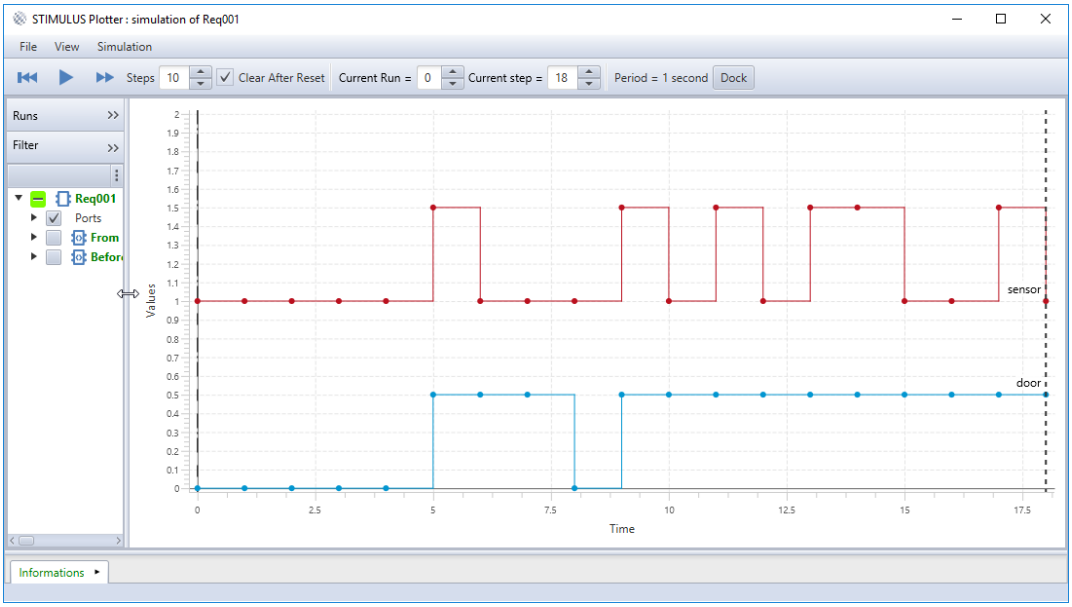


*Figure 18. Seventh simulation*

# 4. Un scénario pour le *sensor*

Vous avez pu constater que le problème de la simulation c'est que le `sensor` change trop souvent de valeur (de manière irréaliste). Nous allons donc créer un scénario pour contrôler les *inputs* de la simulation. Créez un nouveau système dans le *package* `Test`, et nommez-le `SensorInput`.

Ajoutez comme précédemment le glossaire pour ajouter un port `sensor`. Précisez la direction `Out` pour ce port comme illustré en Figure 19

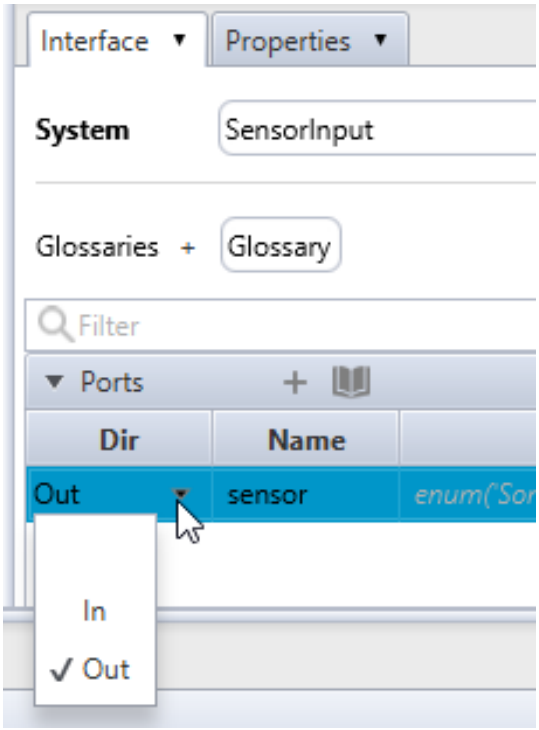


*Figure 19. Port's direction to Out*

Pour écrire le scenario, nous allons utiliser le concept Stimulus d'Automate (ou machine à état).

Les automates peuvent aussi être utilisés pour écrire les exigences. Stimulus ne fait aucune différence entre "requirements" et "scenarios". Pour ajouter un Automate, faites un click droit puis **Insert › Automaton** (ou bien utilisez l'icône correspondante dans la barre d'icônes).

Un nouvel automate apparaît avec l'état initial "State0" comme illustré en Figure 20.

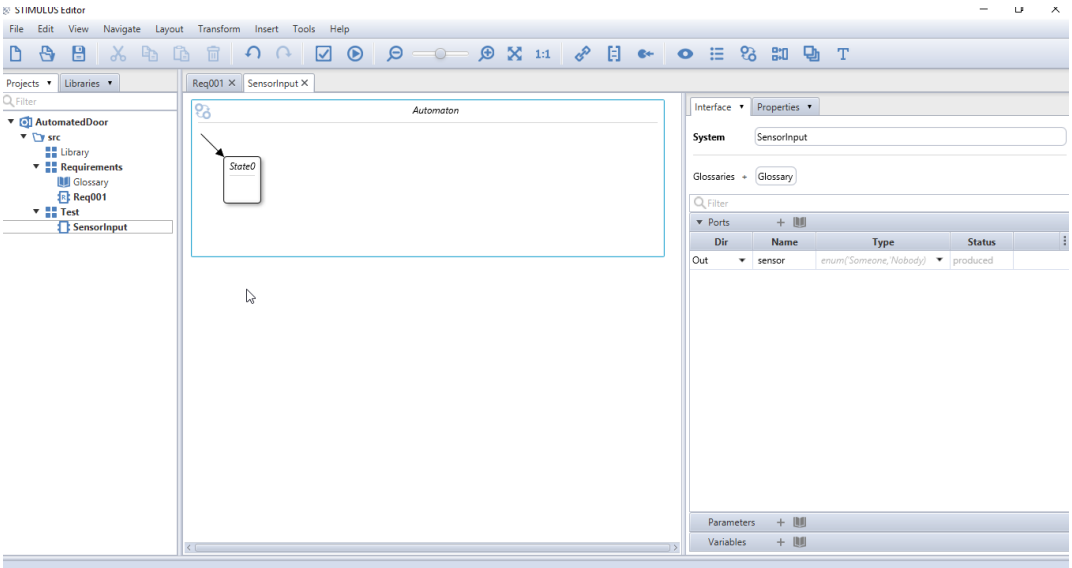


*Figure 20. Automaton created*

Pour créer un autre état, placez la souris près du bord bas de l'état, jusqu'à ce qu'un petit triangle orange apparaisse, puis double cliquez dessus (cf. Figure 21).

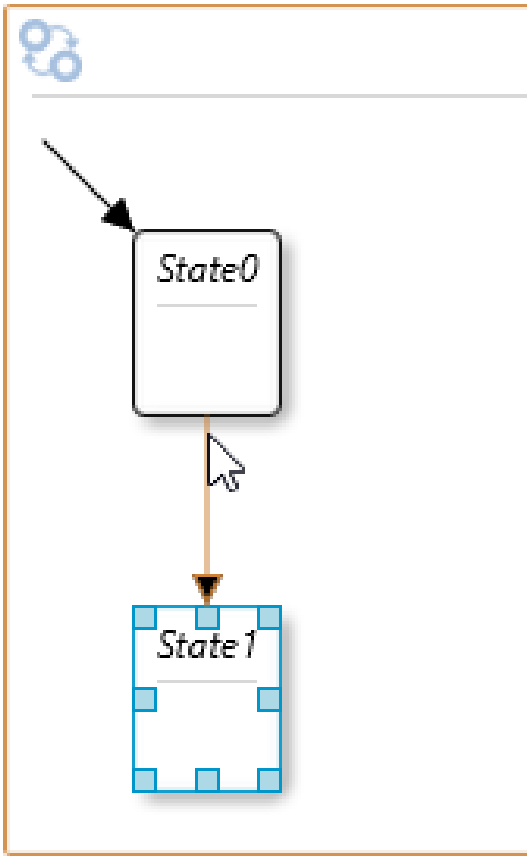


*Figure 21. Two connected states*

Utilisez les mêmes techniques que précédemment pour établir le scénario illustré en Figure 22.

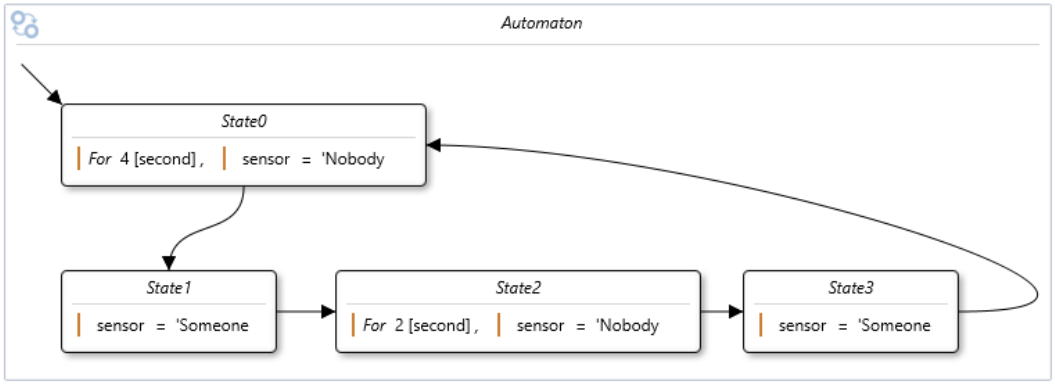


*Figure 22. Writing scenario*

Contrôlez l'exécution du scénario (cf. Figure 23).

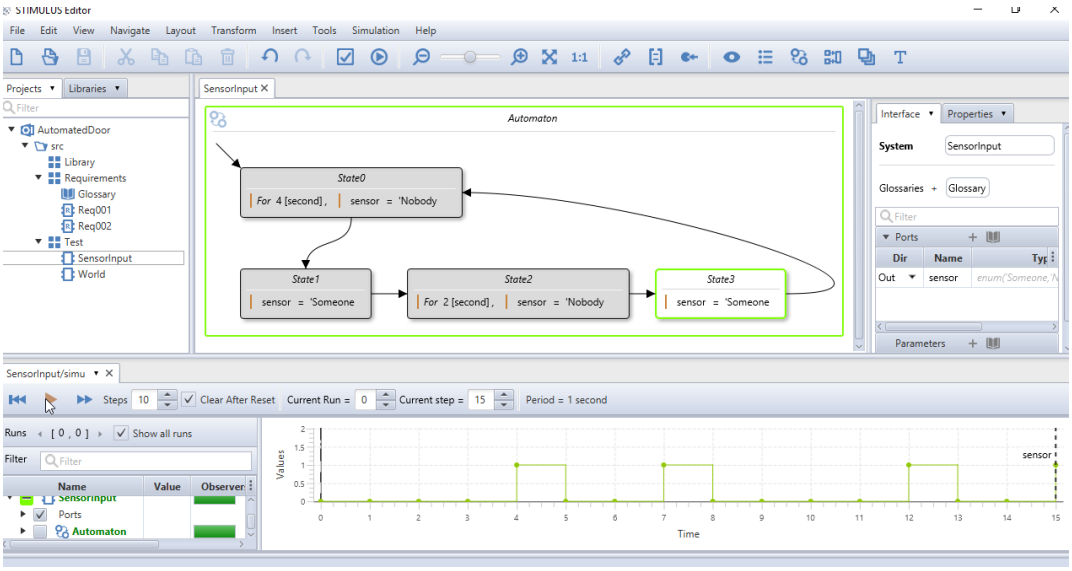


*Figure 23. Scenario Input behaviour*

# 5. Test de la porte

Pour tester la porte, nous allons créer un système haut-niveau en utilisant un *block diagram* (à la SysML), pour connecter la porte et le scenario.

Créez un système appelé "World" dans le *package* `Test`.

Pour ajouter un *block diagram*, faites un click droit puis **Insert › Block Diagram** (ou bien utilisez l'icône correspondante dans la barre d'icônes).

Insérer le block `Req001` en le glissant dans le *block diagram*. Faites de même avec `SensorInput`. Connectez les ports `sensor`.

Pour visualiser les signaux entrants, nous auront besoin d'une sonde. Pour cela, sélectionnez `Req001`, puis **Transform › Connect Ports to Interface** pour obtenir la Figure 24.

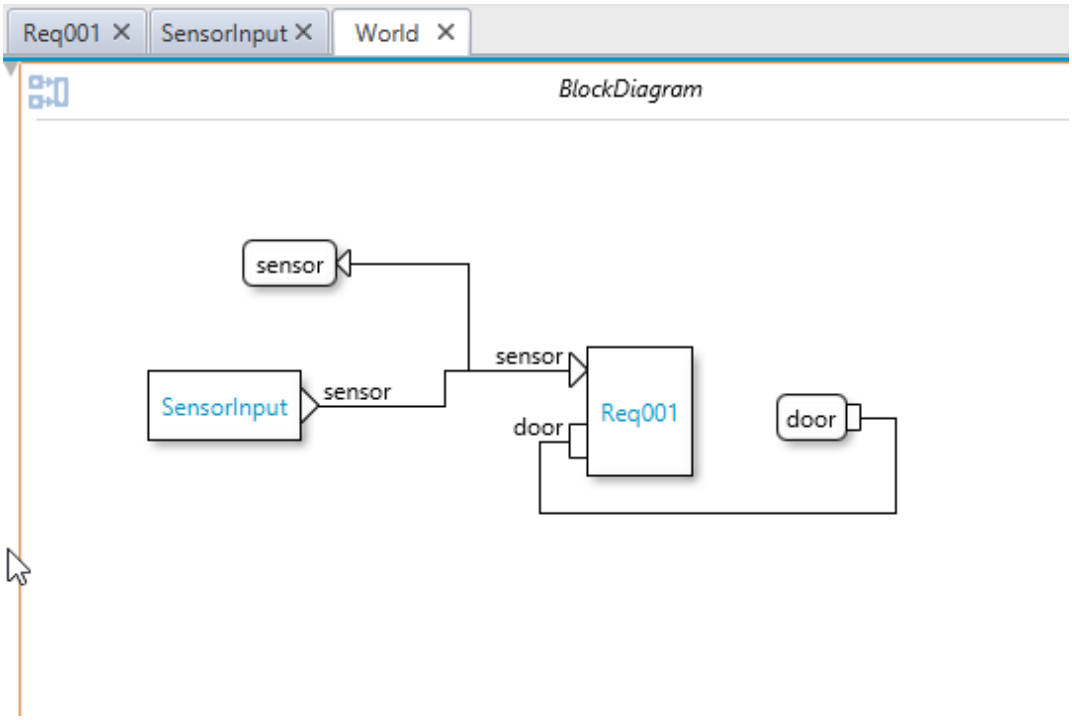


*Figure 24. Probed block diagram*

Enfin exécutez le test comme illustré en Figure 25

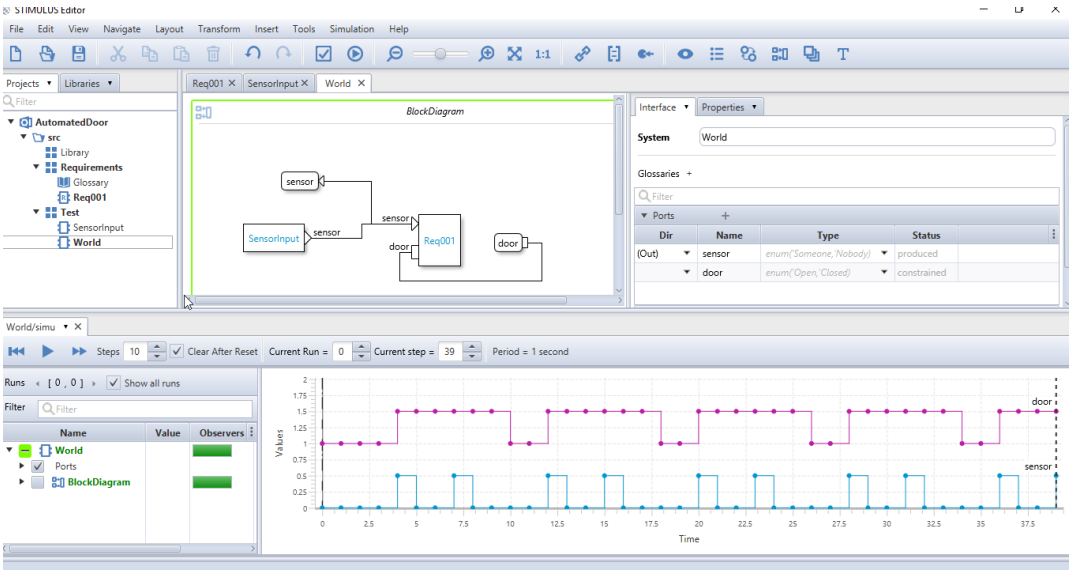


*Figure 25. World simulation*

# 6. Observateurs

Pour renforcer la qualité des exigences, nous recommandons de les vérifier au fur et à mesure de leur écriture. Stimulus permet de transformer les exigences en *Observers* qui vont vérifier les exigences courantes. Cela équivaut à faire des tests de non régression. Un exemple de propriété désirable pour notre porte est que nous voulons garantir que quand quelqu'un est détecté, la porte ne doit pas être fermée.

Pour créer un *Observer*, utilisez l'icône correspondante dans la barre d'icônes (cf. Figure 26).

Renommez le en `Safety` après avoir double cliqué dessus. Puis saisissez l'invariant initial (quand

une personne est détectée la porte ne doit pas être femrée).

Réalisez une exécution comme illustré en Figure 26.

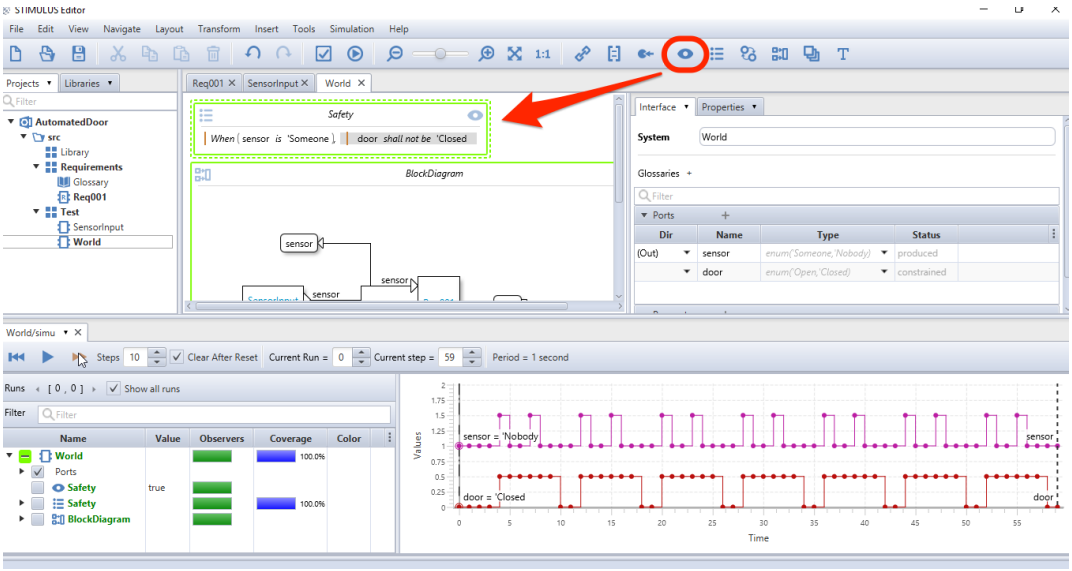


*Figure 26. Simulation with the Observer*

Observez que tout se passe bien de son point de vue (couleur verte) en simulant quelques pas d'exécutions comme illustré en Figure 27.

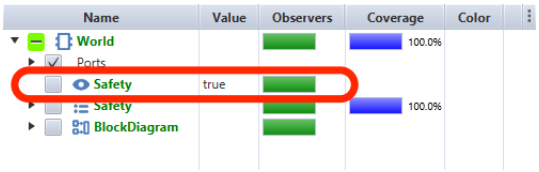


*Figure 27. Observer in plotter*

Reprenez les exigences erronnées du début et vérifiez que l'observateur joue son rôle comme illustré en Figure 28.

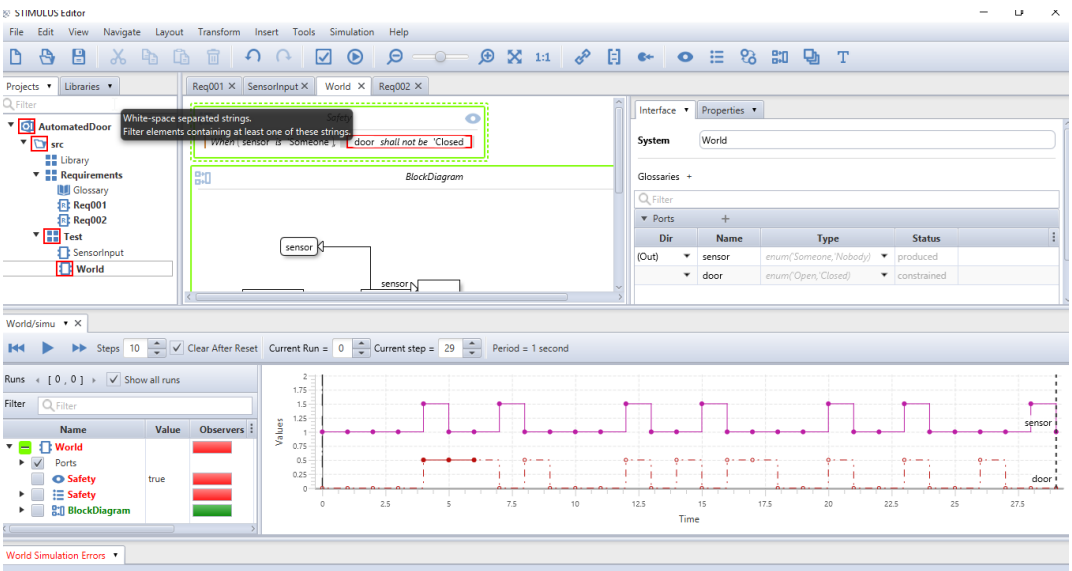


*Figure 28. Violated Observer*

# 7. Basic notions

## 7.1. Predefined, customizable templates

## 7.2. Composition

## 7.3. Refinement of requirement

## 7.4. Observers

High level requirements can be transformed as `observers`.

# 8. Advanced notions

Work in progress...

## 8.1. Glossaries

You can host the main definitions in a `Glossary`. To start one, click on **New › glossary**

You can add the Glossary to the `Interface` (by dragging the glossary close to the little `+` close to `Glossaries`) and then add the ports to your glossary by a clicking on them (see Figure 29).

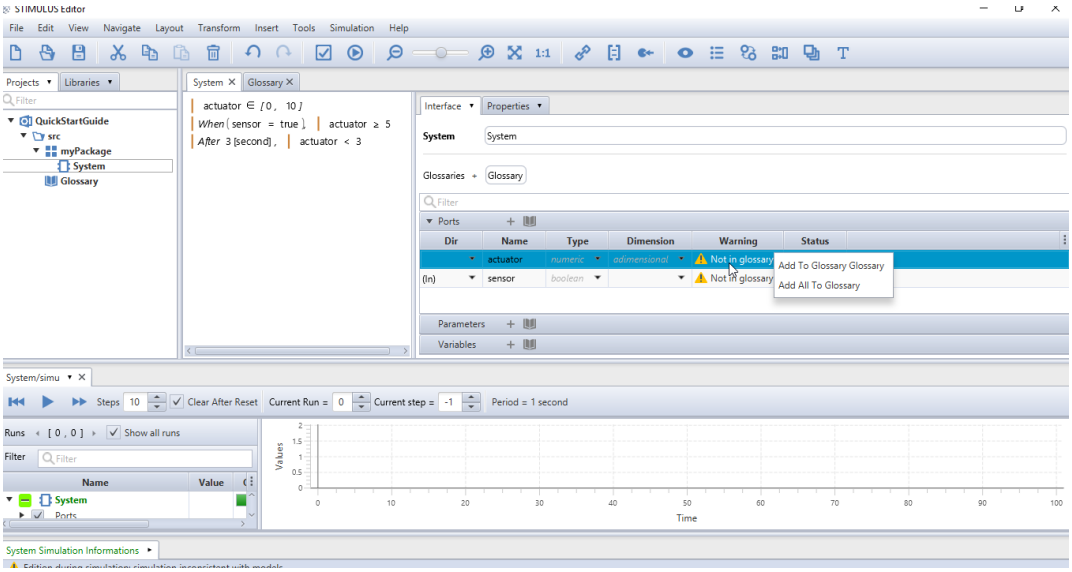


*Figure 29. Adding ports to a glossary*

It can be a good idea to start with the glossary

# 9. FAQ

# 9.1. Where can I find more material ?

https://www.argosim.com/free-trial/

# 9.2. What are the different simulation parameters ?

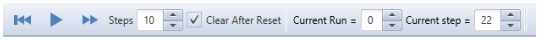


*Figure 30. SimulationBar*

- The **[ simulate one step ]** button performs one simulation step in the plot window.
- The **[ fast forward button ]** performs N simulation steps. The number of steps is defined by the field at the right of the button.
- The **[ reset button ]** starts a new simulation. When the **[ Clear on reset ]** checkbox is unchecked, the next simulation will be overlaid on the previous one.
- The **[ Period ]** label displays the simulation sampling period. You can change this setting before a simulation in the **[ Properties ]** panel of your system. Insert a simulation period using a dimension (e.g., *1 [second]* or *10 [millisecond]*).

# 9.3. How can I find a function or language construct ?

Select the `Library` folder and use the seach bar. Figure 31 illustrates the search for the `when` operator.

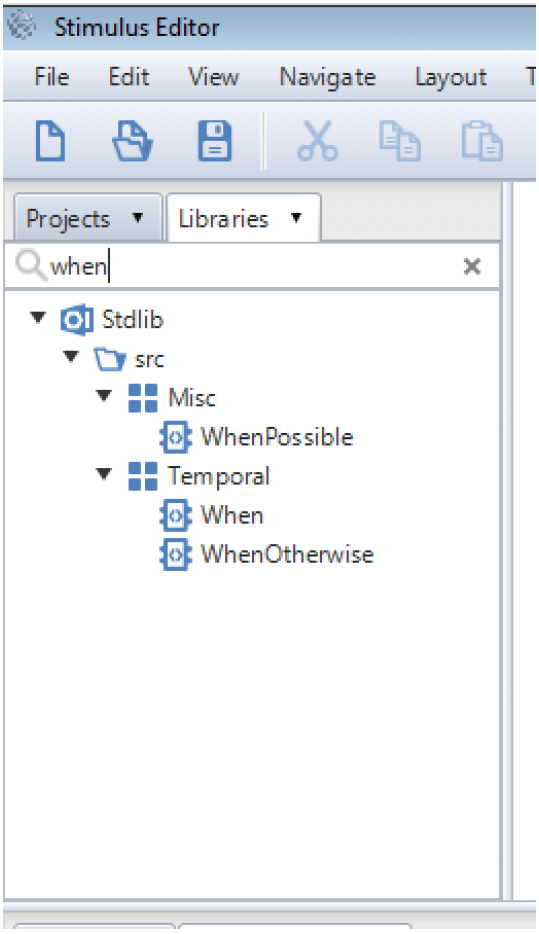


*Figure 31. Search tool*

## 9.4. How can I move a port on a system ?

## 9.5. Comment déplacer graphiquement le port d’un système ?

Pour déplacer un port dans un bloc diagramme, maintenez la touche btn:MAJ[] (Shift) pendant que vous vous déplacez avec clic gauche de la souris.