

## 1 SimioAgentLibrary Installation Manual for End Users

The “SimioAgentLibraryInstaller.msi” installer is provided to facilitate the basic installation of the AgentLibrary (see Appendix E) and installs all the necessary components in the following folder:

```
%programfiles(x86)%\Simio\UserExtensions\SimioAgentLibrary
```

After installation, the add-in is available as described in Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**<sup>1</sup>. However, the modeling components still have to be integrated into Simio and this can be done in two ways:

- 1) The AgentLibrary, with the modeling components, is globally integrated into Simio (applies to all projects).
- 2) The AgentLibrary, with the modeling components, is manually integrated for each project.

Global integration procedure:

1. Start Simio
2. The settings dialog can be opened from the main menu via “File -> Settings”. The AgentLibrary is then entered into the “Additional Libraries To Load” setting. To do this, the path

```
%programfiles(x86)%\Simio\UserExtensions\SimioAgentLibrary\Data\AgentLibrary.spfx
```

has to be replaced by the effective path (i.e. without the placeholder %programfiles(x86)%). For instance:

```
C:\Program Files (x86)\Simio\UserExtensions\SimioAgentLibrary\Data\AgentLibrary.spfx
```

see also Figure 1.

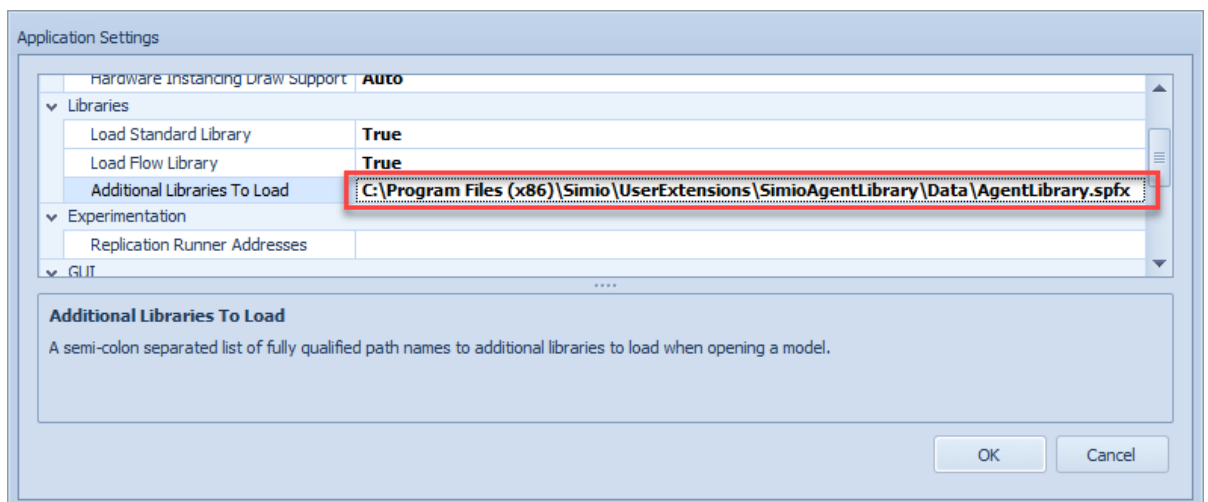


Figure 1: Integrating the SimioAgentLibrary

<sup>1</sup> Note: A licensed version of Simio (not Simio Personal Edition) is required to run the SimioAgentLibrary.

3. Close and restart Simio, whereupon the AgentLibrary should load automatically every time Simio is started (see Figure 2).

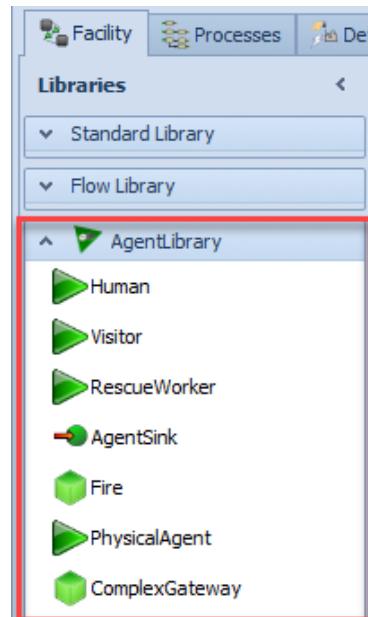


Figure 2: AgentLibrary in Simio

Alternatively, the library can also be manually integrated:

1. Start Simio.
2. Load the library via the Simio main menu ( Figure 3).

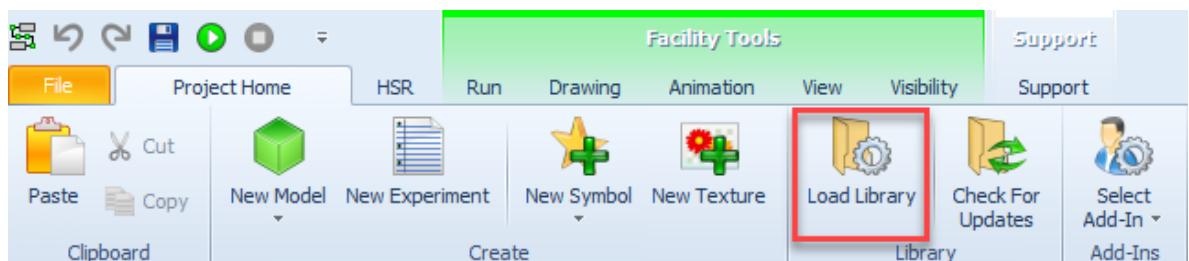


Figure 3: Manually loading the SimioAgentLibrary

The following file or library must be selected in the subsequent file selection dialog:

```
%programfiles(x86)%\Simio\UserExtensions\SimioAgentLibrary\Data\AgentLibrary.spfx
```

3. The library is now made available as described in Figure 2. Please note that the library has to be manually integrated for each new project in accordance with the instructions.

The installer also installs the models used in Chapters 1.1 and **Fehler! Verweisquelle konnte nicht gefunden werden..** These are located under the following path after installation:

```
%programfiles(x86)%\Simio\UserExtensions\SimioAgentLibrary\Data\
```

The AgentLibrary elements can now be used for modeling. The respective state charts can also be modeled for the agents of a model, as described in Chapter **Fehler! Verweisquelle konnte nicht gefunden werden..**

## 1.1 Getting started: Modeling and simulating a consumer market

This chapter provides step-by-step instructions for creating a simple, agent-based model using the SimioAgentLibrary. The example implements a consumer market model, in which each consumer is modeled as an agent and the simulation aims to examine the introduction of a product to the market. Since human (consumer) decisions always include stochastics, agent-based modeling and simulation are ideal for market simulations.

The following assumptions are made:

- The model includes 500 people who do not use the product.
- A combination of word-of-mouth advertising and standard advertising should encourage people to purchase the product.
  - Standard advertising results in 1% of potential consumers per day wanting to purchase the product.
  - With respect to word-of-mouth advertising, on average, an existing consumer speaks to one other potential consumer per day. On average, 1% of potential consumers purchase the product based on word-of-mouth advertising.
- A specific period of time is required to deliver the product to a consumer. This can range between 1 and 25 days, but the average is 2 days.
- A consumer waits an average of 7 days for the product give or take 15% either way, before deciding to cancel the purchase.
- The product service life is 6 months.

The entire model is also available on the project data CD as a Simio project file (see Appendix E – ConsumerMarket.spfx). The model is also installed by the installer (see Chapter 1). The problem statement is based on [10].

### 1.1.1 Step 1: Creating the agent population

Note: It is assumed that the AgentLibrary has been successfully installed as described in Chapter 1 and globally integrated.

1. Start Simio, whereupon a blank project will be created. Save it under the name “ConsumerMarket”.
2. Switch to the model’s facility window.
3. Now create a new source (standard library element) and call it “ConsumerSource”.
4. Next, create a “Human”-type agent object and a complex gateway and label both elements as described in Figure 4. The ConsumerSource output node is connected to the start gateway’s input node using a path object (standard library).

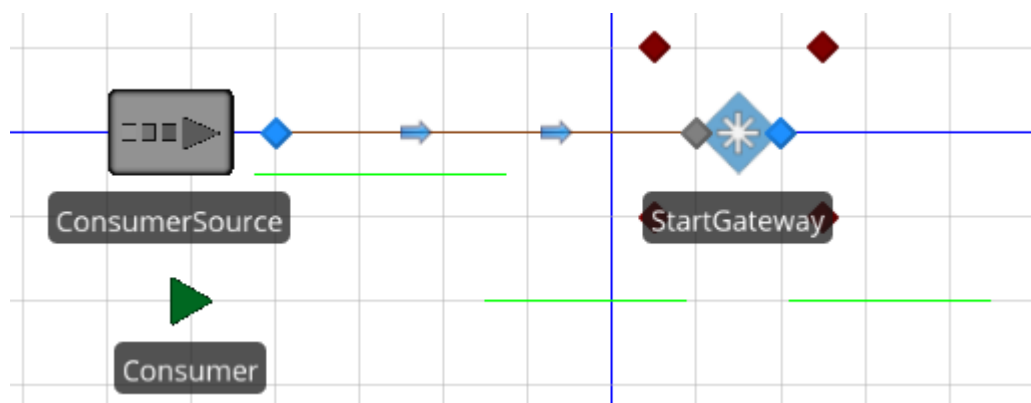


Figure 4: Base model for a consumer market

- Three symbols are then assigned to the consumer object to visualize the state of the agents during the simulation. To do this, select the consumer object in the facility window and add two additional symbols using the “Add Additional Symbol” function (“Symbols” option on the main menu). Finally, the individual symbols are assigned different colors (see Figure 5).

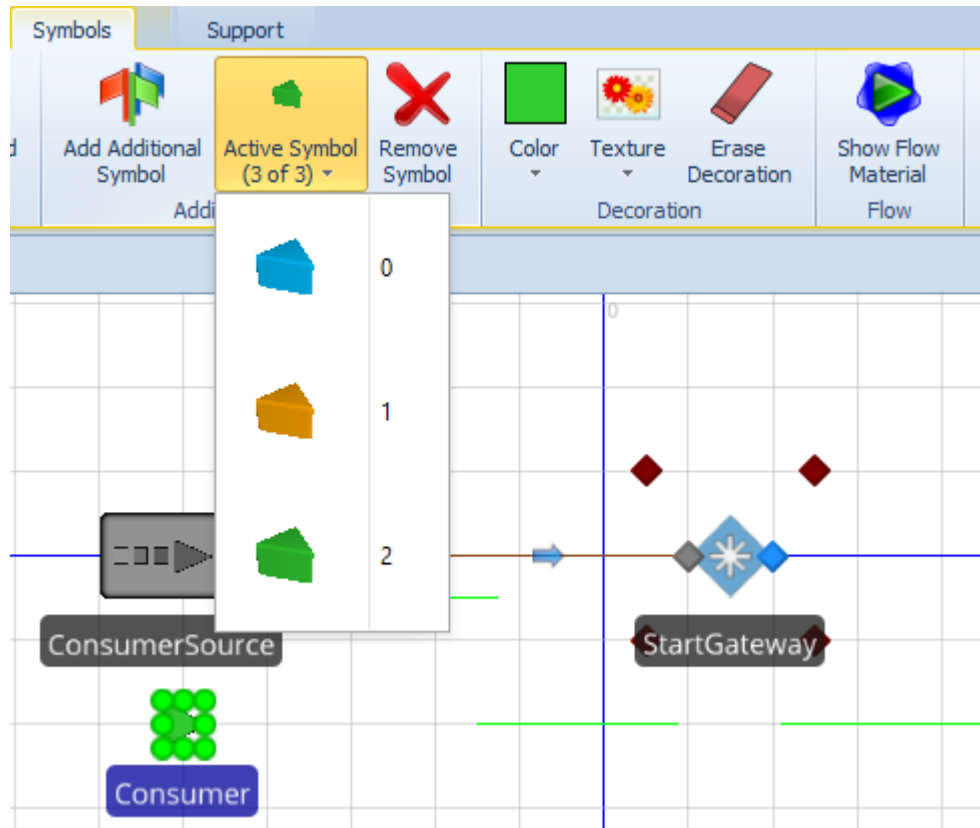


Figure 5: Consumer object with different symbols and colors

A Simio state variable should be created to assign the current symbol. Switch to “Definitions”, select the “States” section from the same, then create a new “String”-type state variable and call it “ConsumerSymbol”. The symbol is calculated in the consumer object’s “Current Symbol Index” property using the following Simio expression:

```
Math.If(String.Length(ConsumerSymbol) <= Human.ID, 0,
String.ToReal(String.Substring(ConsumerSymbol, Human.ID, 1)))
```

Add this (single-line) to the “Current Symbol Index” property (see Figure 6).

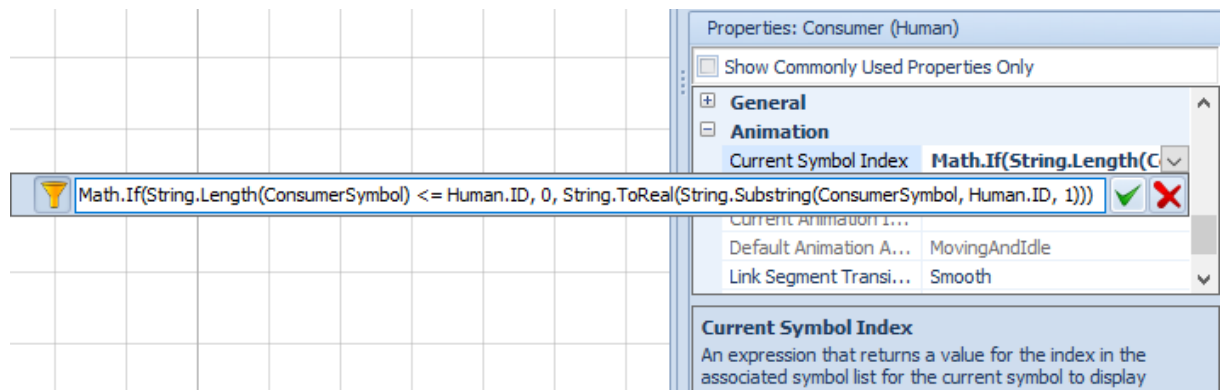


Figure 6: Calculation of the “Current Symbol Index” for the consumer object

6. The number of consumers Simio is supposed to create when starting the simulation must also be configured. According to the assumption, a population of 500 consumers is to be simulated. To do this, select the ConsumerSource object and configure it as follows:
  - Entity Type = Consumer
  - Entities Per Arrival = 500
  - Maximum Arrivals = 500
  - Interarrival Time = Infinity

With this setting, 500 consumer-type entities are created immediately when starting the simulation.

## 1.1.2 Step 2: Define consumer behavior

The next step involves us modeling the agents’ behavior, for which a state chart is created for the consumer object.

1. Start the agent management interface from the Simio management via “HSR -> Simio AgentLibrary” (Fehler! Verweisquelle konnte nicht gefunden werden.). Subsequently, the existing model will be displayed with the “Consumer” agent in the open agent management interface (Figure 7).

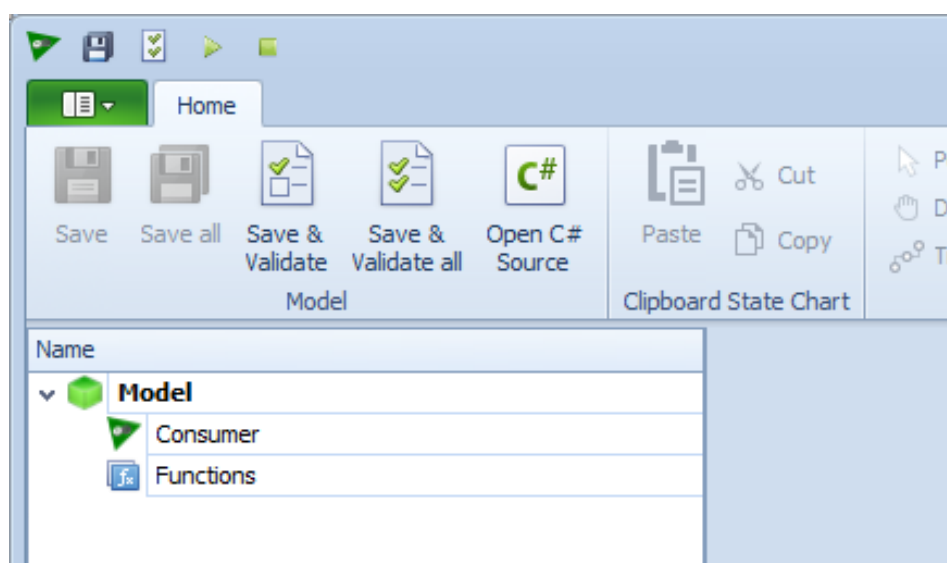


Figure 7: Agent management interface with model and “Consumer” agent

After double-clicking on “Consumer”, the state chart designer opens, allowing the state chart for the agent to be modeled.

2. A consumer can take on three different states:
  - a. Potential consumer
  - b. A consumer having opted to purchase, who is awaiting the product
  - c. Consumer

Accordingly, we model the three states, label them and assign each a color (as described in Figure 8). The colors should correspond to the respective consumer object colors in Figure 5.



Figure 8: Consumer states

3. Whenever a consumer is created, it should initially be assigned the “Potential Consumer” state and be randomly positioned within a defined area of the facility window. For this area, we define the box  $\begin{pmatrix} 10 \\ 0 \end{pmatrix}$  to  $\begin{pmatrix} 40 \\ 30 \end{pmatrix}$  in the Simio facility window (see Figure 10). To position the consumer in this area, we create an entry point in the state chart designer which points toward the “Potential Consumer” state and label it with “Set position” (Figure 18). To assign the initial position, we create a new “SetInitialPosition” function (menu group “Functions -> Add”) and add it as an action for the entry point (Figure 9).

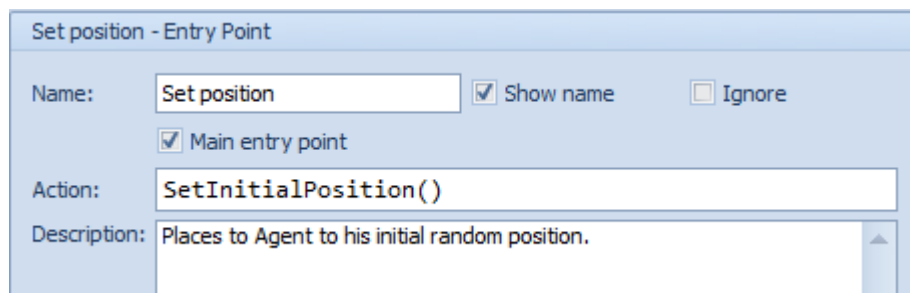


Figure 9: Entry point configuration

The code for the “SetInitialPosition” can be found in Listing 1.

```
var x = Random.GetUniform(10.0,40.0);
var y = Random.GetUniform(0.0,30.0);
```

```
Agent.JumpTo(new Vector2(x,y));
```

Listing 1: SetInitialPosition function

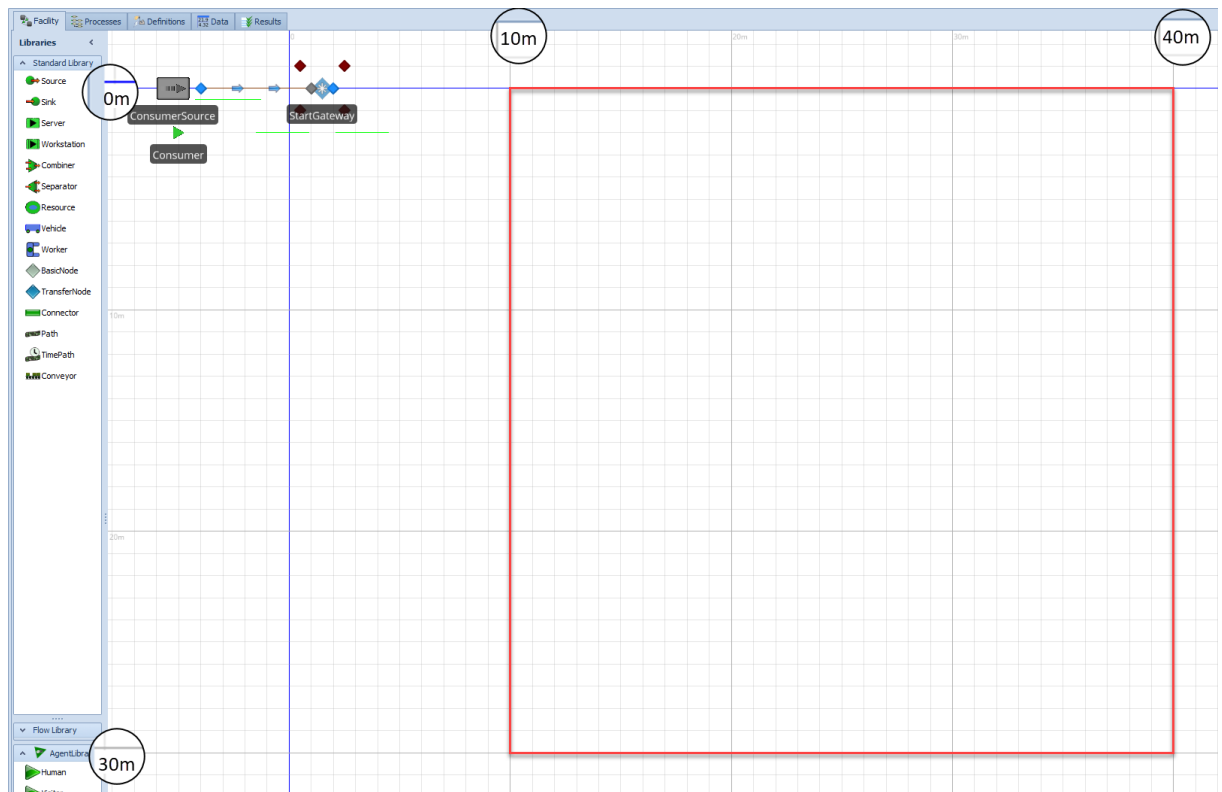


Figure 10: Consumer agent region in the facility window

To use the facility window coordinate system to position the agent, it has to be configured in the “StartGateway” object. To do this, set the “StartGateway” “FacilityName” property in the Simio expression “Model” (see Figure 11).

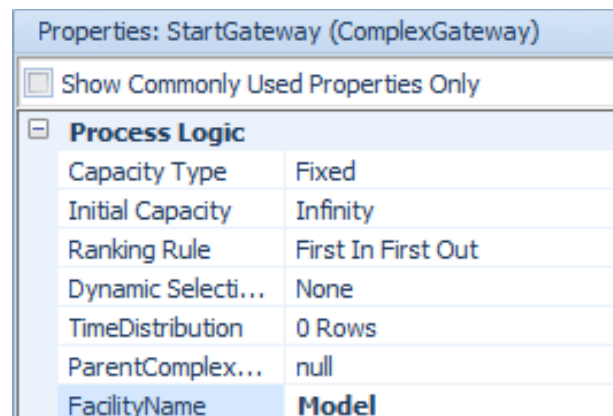


Figure 11: FacilityName for StartGateway

- The assumption that 1% of potential consumers will purchase the product daily should be modeled in the next step, reflecting which, we add a transition between “Potential Consumer” and “WantsToBuy”, which we call “Ad” (Figure 18). It is a “Rate”-type transition; assigned a rate of 0.01 per day, which corresponds to a distribution of 1% per day (Figure 12). The rate type corresponds to an exponential distribution with the average value of “rate”.  
The impact of “word-of-mouth advertising” is modeled in a later step.

Figure 12: “Ad” transition configuration

5. A consumer waits an average of 7 days to receive the product, give or take 15% either way, before otherwise deciding to cancel the purchase. To implement this, we add a “Timeout”-type transition between “WantsToBuy” and “Potential Consumer”, which we call “CantWait”. To model this behavior, we use a triangular distribution with an average value of 7 and a deviation of 0.15 (Figure 13).

Figure 13: “CantWait” transition configuration

6. Next, the product delivery time is modeled; lasting between 1 and 25 days and with an average value of 2 days. For this aspect, we can use a “Timeout”-type transition, which is added between “WantsToBuy” and “Consumer” and called “Purchase”. Since we know the minimum, maximum and average values, we can use the normal triangular distribution (Figure 14).



Purchase - Transition	
Name:	Purchase <input checked="" type="checkbox"/> Show name <input type="checkbox"/> Ignore
Triggered by:	Timeout
Timeout:	Random.GetTriangular(1, 25, 2) days
Action:	
Guard:	
Description:	Let's assume it typically takes a user two days to get the product. This means once the consumer's statechart enters the state WantsToBuy, it will proceed to the state User with a two-day delay.

Figure 14: "Purchase" transition configuration

7. The product has a service life of 6 months. A "Timeout"-type transition, which is also used for this aspect, is placed between "Consumer" and "WantsToBuy" and we call this "Discard". We configure 6 months as the timeout value (Figure 15).

Discard - Transition	
Name:	Discard <input checked="" type="checkbox"/> Show name <input type="checkbox"/> Ignore
Triggered by:	Timeout
Timeout:	6 months
Action:	
Guard:	
Description:	DiscardTime = 6 Define the Products lifespan

Figure 15: "Discard" transition configuration

8. Now, the "word-of-mouth advertising" requirement is still missing. A consumer should have contact with one other agent per day. We achieve this with an internal "Rate"-type transition in the "Consumer" state, which we call "Contact". As the action, the "Buy" message should be sent to a random agent (Figure 16).

Contact - Transition	
Name:	Contact <input checked="" type="checkbox"/> Show name <input type="checkbox"/> Ignore
Triggered by:	Rate <input type="checkbox"/> Internal transition
Rate:	1 per day
Action:	Agent.SendToRandom("Buy")
Guard:	
Description:	Contact one Agent per Day and send him the Message "Buy" to initiate Word-of-Mouth-Propaganda.

Figure 16: "Contact" transition configuration

1% of potential consumers react to the word-of-mouth advertising. We achieve this with a “Message”-type transition between “Potential Consumer” and “WantsToBuy”, which we call “WoM”. The transition reacts to the receipt of the “Buy” message, but only in 1% of cases (see Figure 17).

WoM - Transition

Name:

WoM

☒ Show name

☐ Ignore

Triggered by:

Message

Message type:

string

Fire transition:

☐ Unconditionally
 ☒ On particular message
 ☐ If expression is true

Message:

"Buy"

Action:

Guard:

Random.RandomTrue(0.01)

Description:

AdoptionFraction = 0.01  
  
 To define a person's influence on others, a number that we'll express as the percentage of people who will use the product after they come into contact with the consumer.

Figure 17: “WoM” transition configuration

The entire state diagram can be found in Figure 18.



Figure 18: Entire state diagram for the consumer agent

## 1.1.3 Step 3: Add diagram to visualize the result

Two aspects are implemented to visualize the result:

- 1) The agents in the facility window are color-coded in the respective state's color
- 2) The number of agents per state is displayed on a status plot diagram

To achieve this, a Simio state variable for each state, including the number of respective agents, has to be created in Simio:

1. Switch to Simio. Select the "Definitions" section, then "States". Now create three "Integer"-type state variables:
  - NumberPotential
  - NumberWantsToBuy
  - NumberConsumers
2. The Simio state variables now have to be added/subtracted as soon as an agent enters/exits the corresponding state on the state chart. To do this, switch back to the state chart designer. For each state, the corresponding state variable should be increased by 1 or reduced by 1 in the entry and exit actions respectively. In addition, we want to assign the corresponding color to the agent in the entry action. Since the action comprises two expressions, we create a function for each state, which is retrievable in the entry action, whereupon the following functions are created:
  - EntryPotentialConsumerState
  - EntryWantsToBuyState
  - EntryConsumerState

The functions are each assigned the code as specified in Listing 2 to Listing 4.

```
SetStateValue("NumberPotential", GetStateValue<int>("NumberPotential")+1);  
SetConsumerSymbol(0);
```

**Listing 2: EntryPotentialConsumerState function**

```
SetStateValue("NumberWantsToBuy", GetStateValue<int>("NumberWantsToBuy")+1);  
SetConsumerSymbol(1);
```

**Listing 3: EntryWantsToBuyState function**

```
SetStateValue("NumberConsumers", GetStateValue<int>("NumberConsumers")+1);  
SetConsumerSymbol (2);
```

**Listing 4: EntryConsumerState function**

The "EntryXXXState" functions can now be integrated into the entry actions for the relevant states in the state chart designer, as shown in Figure 19 with the "Potential Consumer" state as an example.

3. In order for the Simio state variables to be decremented by an agent when exiting the state, the corresponding logic has to be entered in the exit action (see Listing 5 and Figure 19).

```
SetStateValue("NumberPotential", GetStateValue<int>("NumberPotential")-1)
```

**Listing 5: EntryConsumerState function**

Figure 19: Entry action and exit action for the “Potential Consumer” state

4. In addition, the “SetConsumerSymbol” function has to be created. It manipulates the Simio “ConsumerSymbol” state variable which, in turn, acts as a basis for calculating the symbols for the agents. The function contains an “int”-type parameter (see Figure 20).

Name	Type
symbolID	int

Figure 20: “SetConsumerSymbol” function parameters

The code for the “SetConsumerSymbol” function displays is shown in Listing 6.

```
var symb = GetStateValue<string>("ConsumerSymbol");
symb = (symb??'').PadRight(Agent.ID, '0');
var arr = symb.ToCharArray();
arr[Agent.ID-1] = symbolID.ToString()[0];
SetStateValue("ConsumerSymbol", new string(arr));
```

Listing 6: SetConsumerSymbol function

The algorithm creates a string (Simio state variable “ConsumerSymbol”). The respective symbol number is saved for each agent at its position in the string (based on `Agent.ID`). This information is then used in the consumer agents to calculate the current symbol (see Figure 6).

The “NumberPotential”, “NumberWantsToBuy” and “NumberConsumers” state variables can now be used as data sources for the status plot diagram.

1. Switch to Simio and add a status plot diagram in the facility window (Simio main menu “Animation”). Select 4 weeks as the “Time Range”. Change the “Additional Expression” setting as shown in Figure 21.

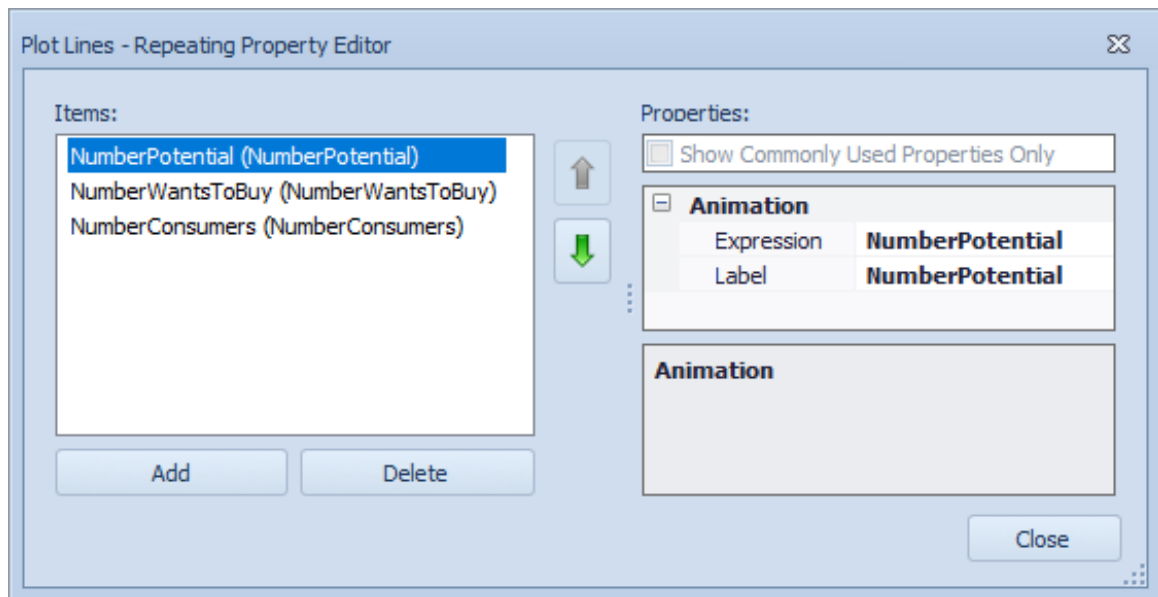


Figure 21: Additional expressions for the status plot diagram

## 1.1.4 Step 4: Executing the simulation

The simulation can now be executed and should cover a period of 20 weeks.

1. In the main menu, under "Run", configure a random starting point and set the end to 20 weeks later (see Figure 22).

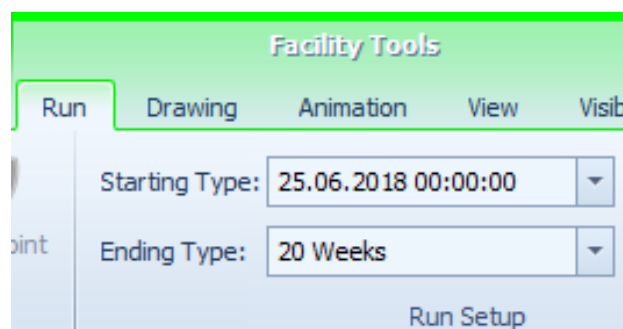


Figure 22: Duration of the simulation

2. Set the "Speed Factor" under "Run" -> "Animation Speed" to 10000.000.
3. To keep the performance high, configure the "Update Time Interval" for the consumer agents to 4 hours (see Figure 23).

Properties: Consumer (Human)	
<input type="checkbox"/> Show Commonly Used Properties Only	
<input checked="" type="checkbox"/> <b>Process Logic</b>	
Initial Health	100
Initial Place	null
Agent Sink	null
FireImpact	20
<input checked="" type="checkbox"/> Steering Behavior	Agent
<input checked="" type="checkbox"/> Update Time Interval	4
Units	Hours

Figure 23: "Update Time Interval" for the consumer agents

- Start the simulation on the agent management interface in the main menu under "Simulation" -> "Run/Pause", which ensures all components are compiled and configured using the current code.

A possible result after a simulation duration of 20 weeks can be found in Figure 24 and **Fehler! Verweisquelle konnte nicht gefunden werden..**

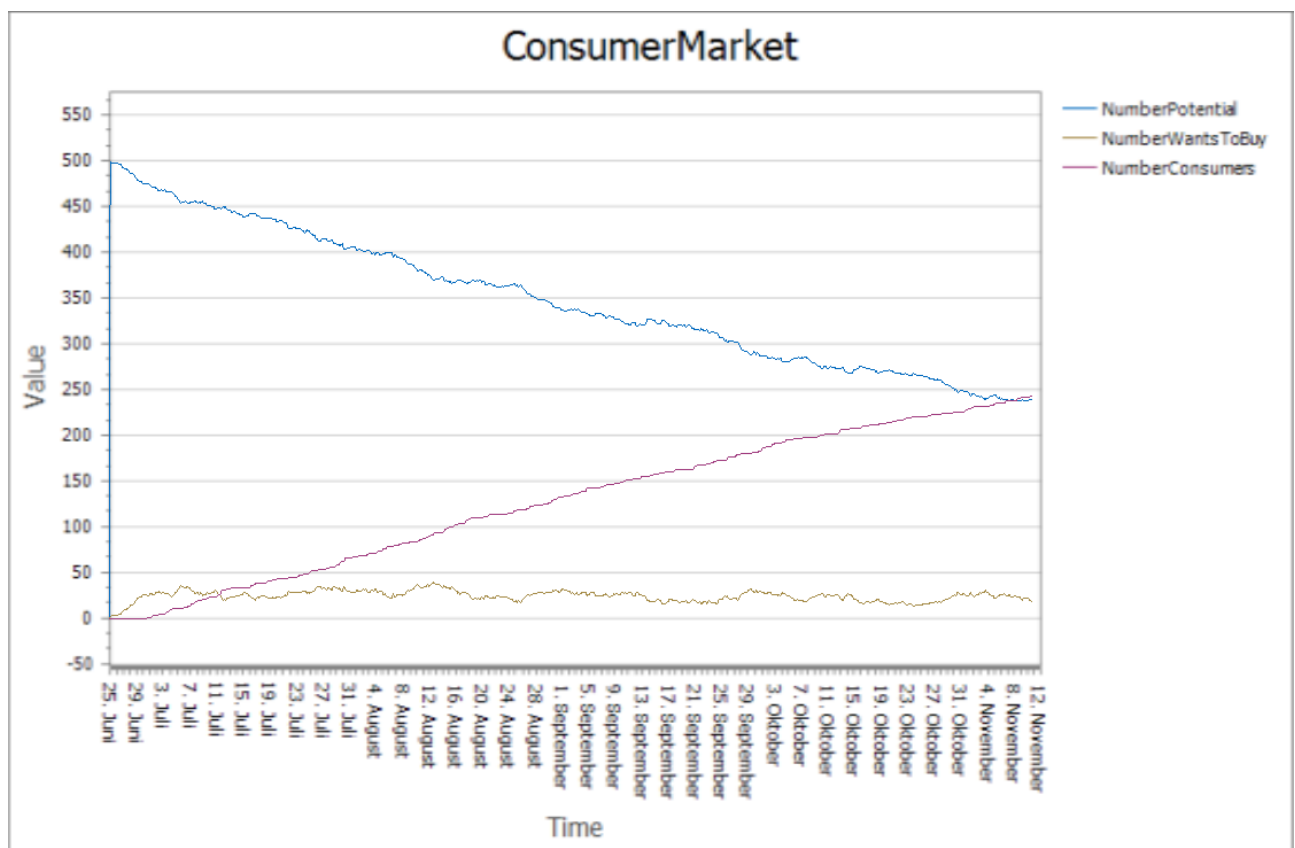


Figure 24: Consumer market result

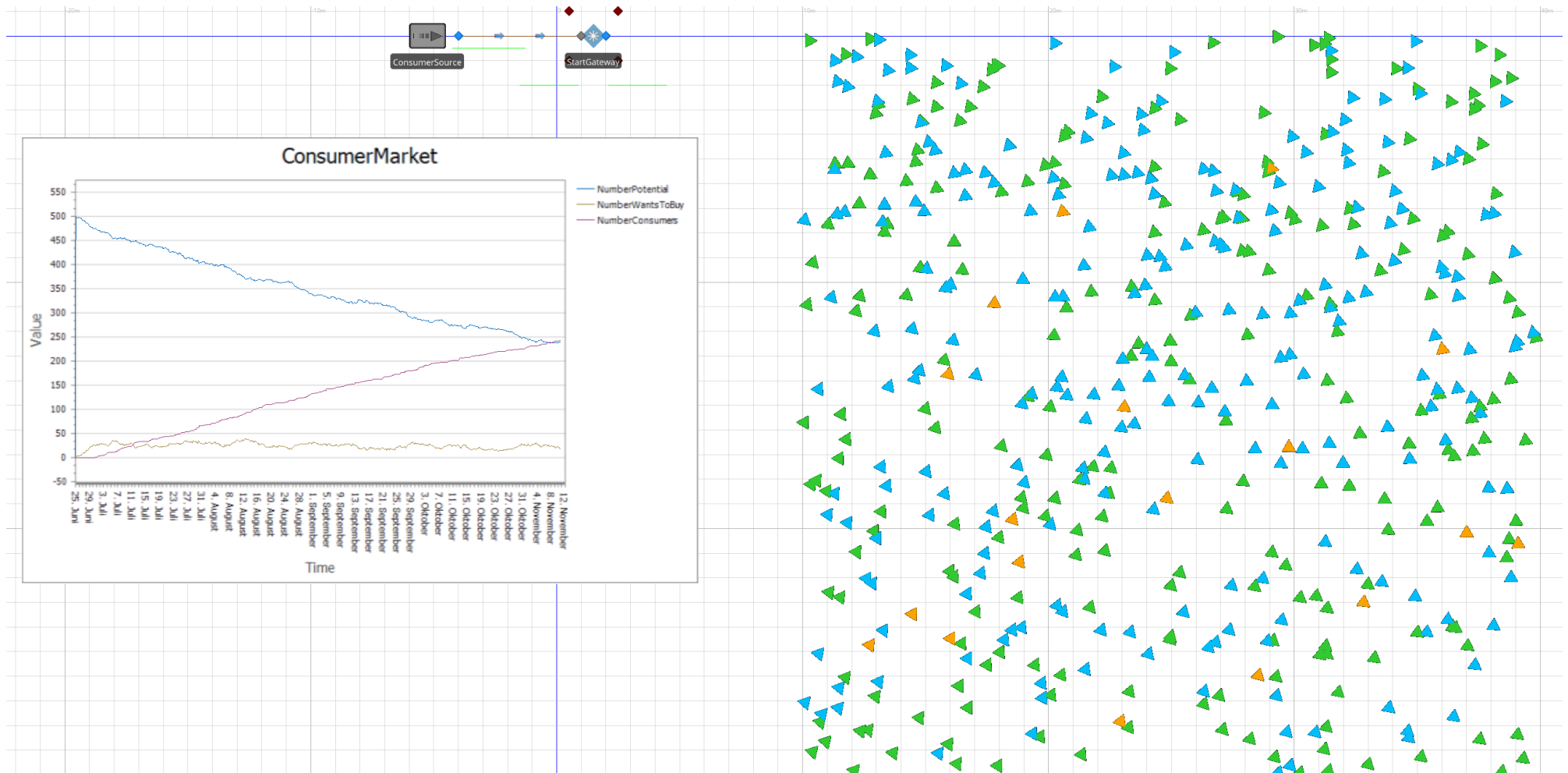


Figure 25: Overview: consumer market result