1.1.1. Trustworthiness Model Editor GE (TWME)

The TWME provides a platform to design new multi-asset system models, comprising of classes and relationships from a generic ontology model. Having created the structure of a new system model, this software allows system designers to generate an ontological representation (in OWL) of the Abstract System Model. Inferences and templates (defined in SPARQL + SPIN) can then be run on this created ontology, which will generate a collection of possible threats to the designed model, as well as class level reasoning for each of the assets in the Abstract System Model.

On starting the system, there is a choice of creating a new model or loading a saved model. Unfinished models are saved in XML instead of OWL/RDF. The reason for this is that a system model in the making might contain inconsistencies, logical errors or missing information and thus be unsuitable for reasoning. Also, the XML file contains information about the display of the model. Once a model is finished, it can be converted to a consistent OWL ontology.

Having set the name for your new Abstract System Model, the editor will display an empty TWME canvas as shown in Figure 2.

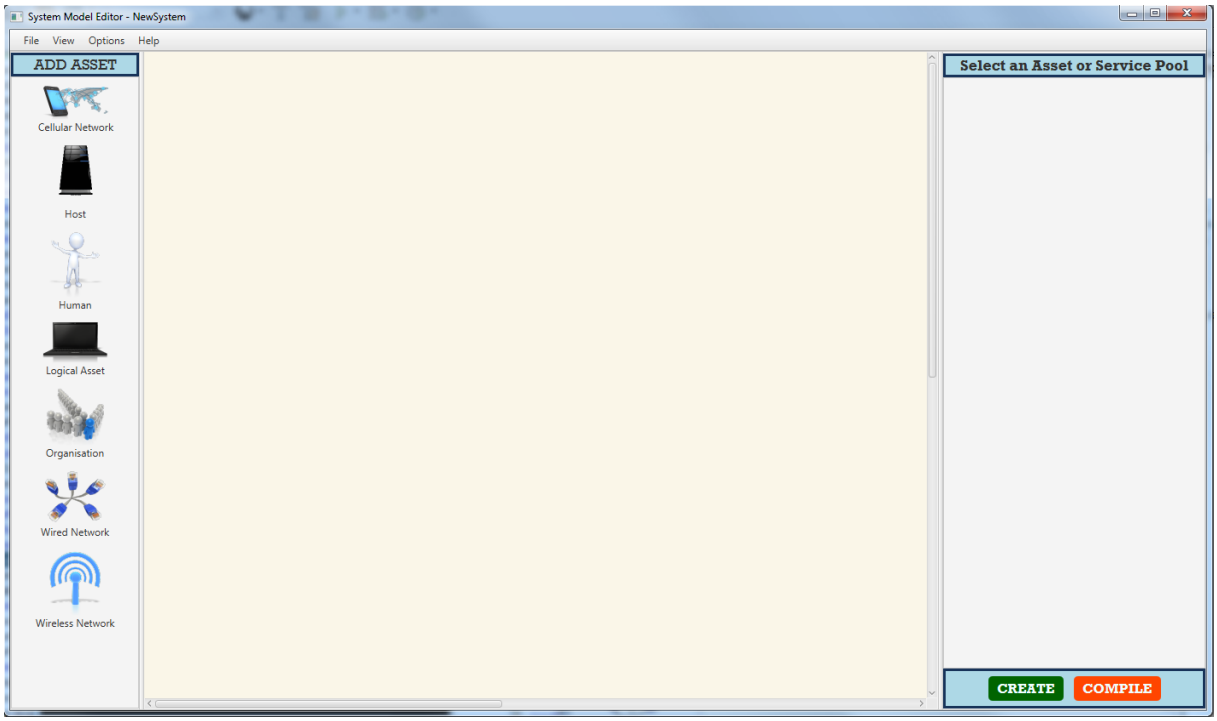


Figure 1: Trustworthiness Model Editor - Empty canvas.

There is a panel on the left hand side which offers the system designer a choice of available asset types. These generic classes can be subclassed by dragging and dropping them into the Trustworthiness Model Editor's canvas. Once dropped, they can be connected to each other where the relation type is read directly from the generic model - only valid relations can be created. Figure 3 shows the process of dragging and dropping an asset onto the empty canvas,

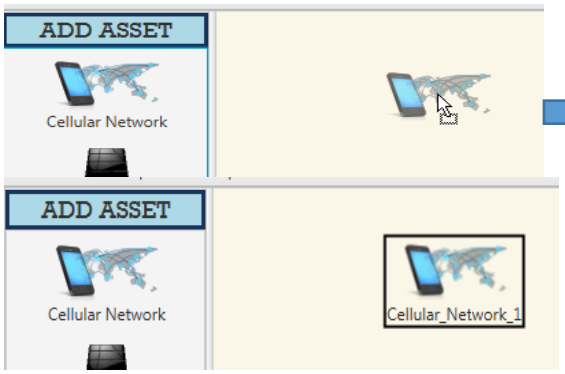


Figure 2: The process of adding an asset.

The available asset types for composing a system model are:

- Cellular Network – a typical mobile phone network
- Host – any physical asset which can host logical assets
- Human – human user/stakeholder/operator
- Logical Asset – processes/software running on the system
- Organisation - company
- Wired Network
- Wireless Network

After adding assets to the Abstract System Model, it is then possible to add relationships between the assets, in order to define the structure of the model. Hovering over an Asset on the canvas reveals its "add relationship from" button:

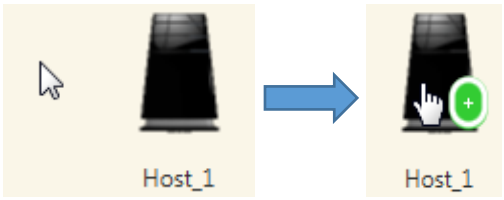
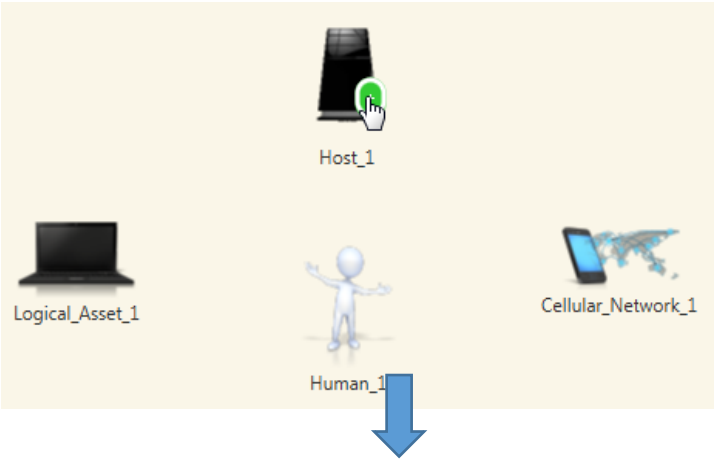


Figure 3: Process of adding relationships between assets.

The possible relationship types are read directly from the OPTET Generic Model. Only valid relationship types will be available and only matching asset types can be selected as relation endpoints. This ensures consistency and prevents errors from being introduced. Figure 4 and Figure 5 show the process of adding a relationship.



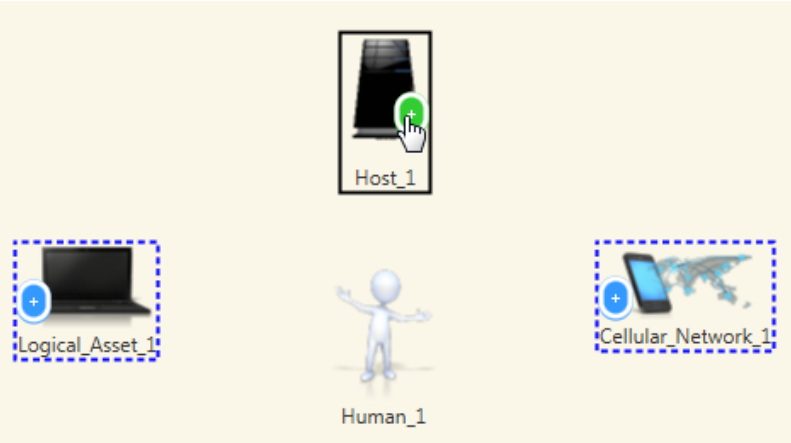


Figure 4: Process of adding relations between assets.

If there is only one type of relationship available for a relationship between the two assets, it is automatically defined and displayed (See Figure 6).

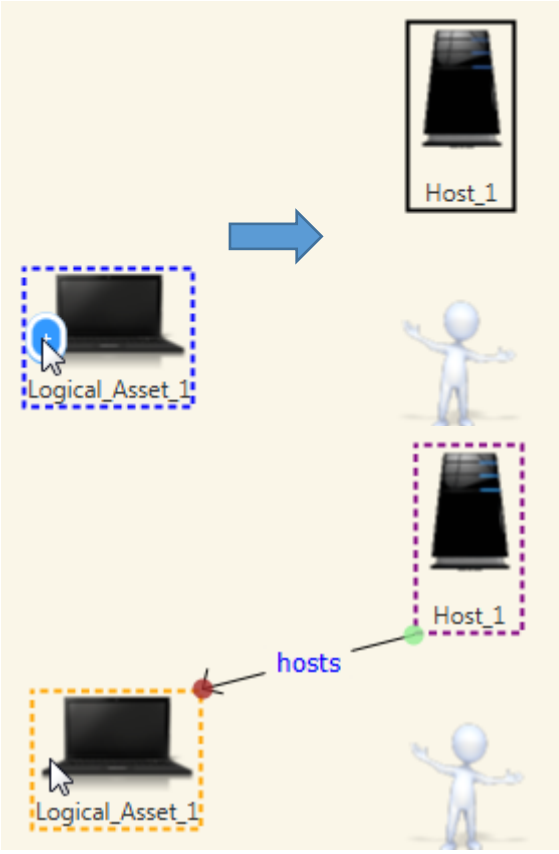


Figure 5: Automatic relationship infererence.

However, if there is more than one type of relationship available for the relationship between the two assets, the relationship is added but it is displayed and stored as "undefined". This is not a valid relationship type and needs further input from the system designer, which is why it is highlighted in red. The correct relationship type can be chosen by clicking the relation and selecting the desired type from the menu (see Figure 7).

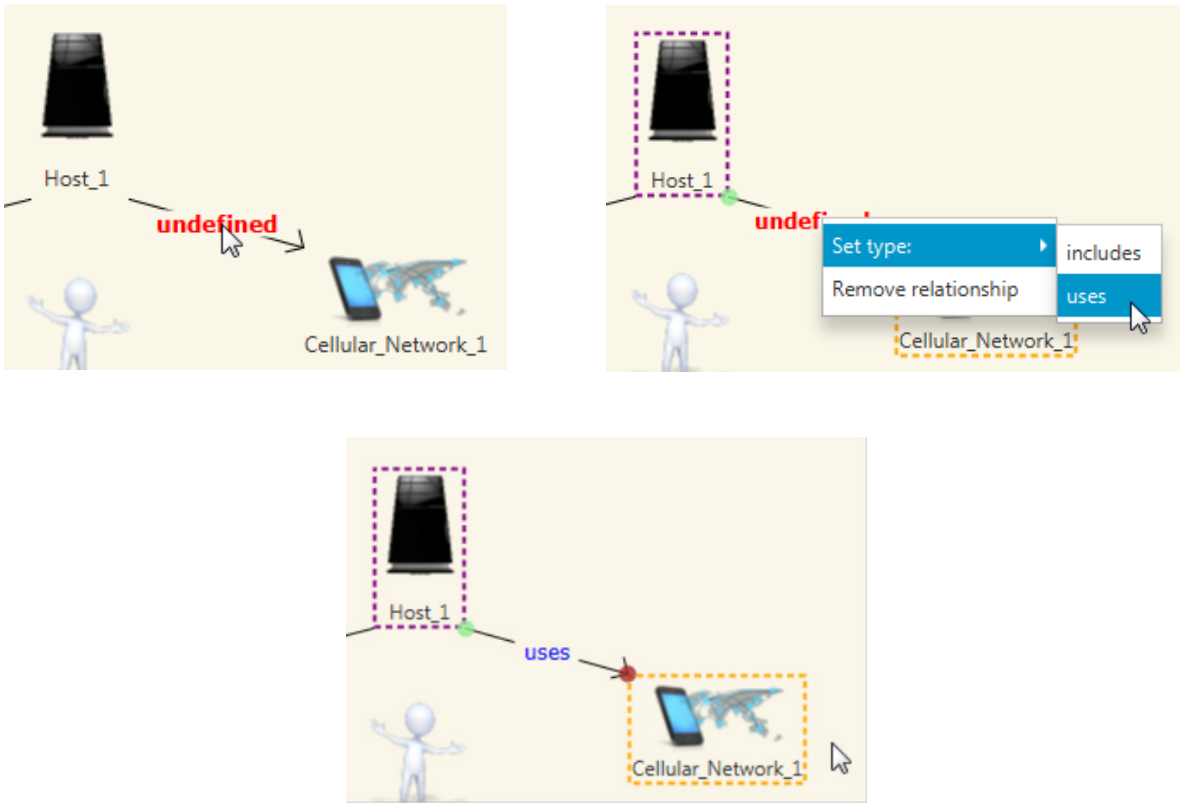


Figure 6: Manually assisted relationship inference.

A service pool is an abstract type of Asset, which cannot be directly added to the Abstract System Model. It describes a service pool for a uses relationship between two logical assets in the client service pattern. When a "uses" relationship is added between two Logical Assets, a service pool is automatically added to this relationship (See Figure 8).

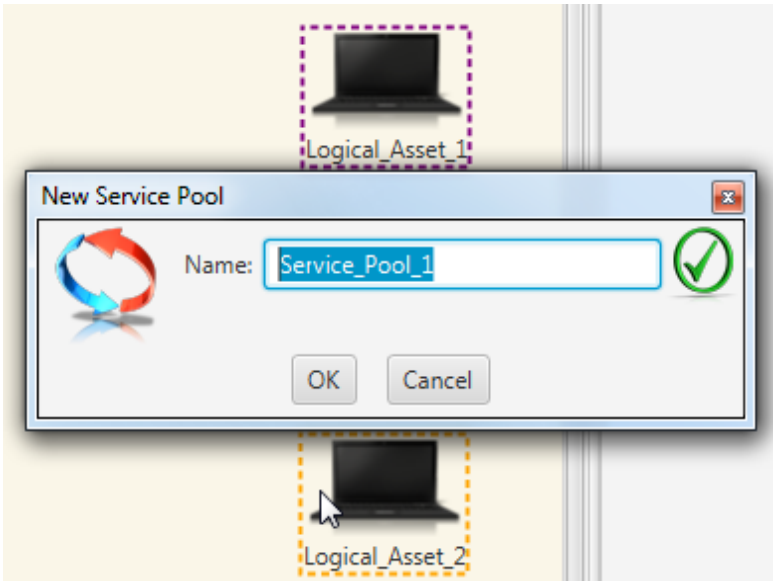


Figure 7: Service Pool addition.

Information for the Service Pool will be displayed in the Information Panel and will be displayed when the uses relationship is selected (See Figure 9).

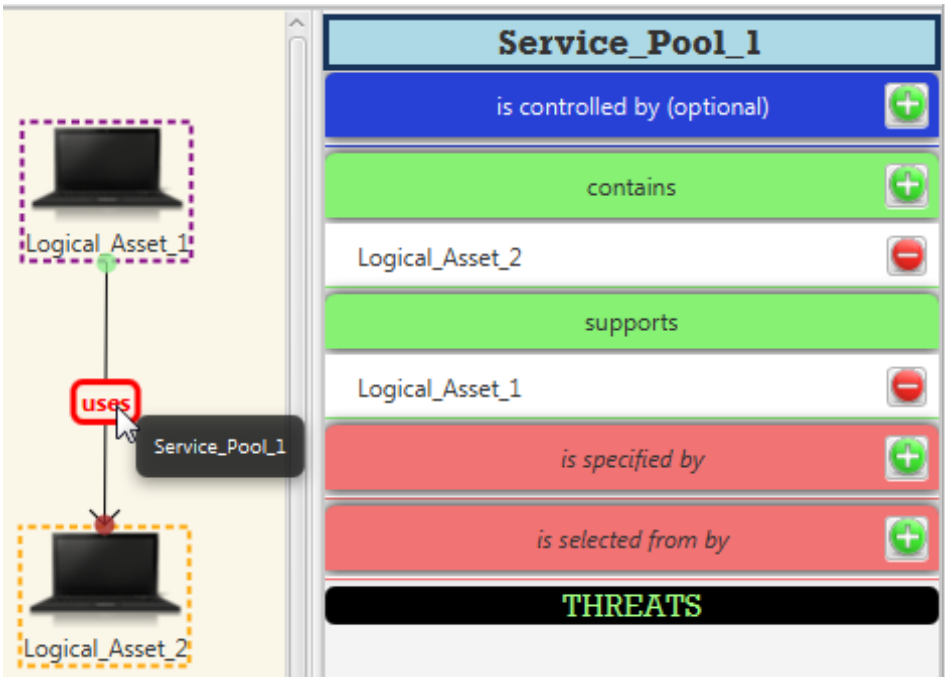


Figure 8: Service Pool configuration.

Each service pool has 4 relationships that must be defined before it can be considered complete, The "contains" and "supports" relationships are automatically defined by the "uses" relationship that created the Service Pool; it always supports the client and contains the service. The two remaining relationships can be completed by selecting one of the possible assets. What assets are valid for these relationships is also knowledge from the Generic Model and its restrictions.

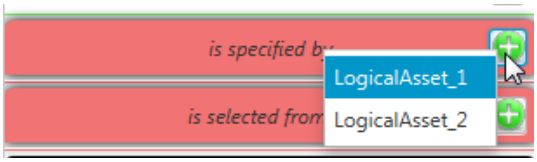


Figure 9: Service Pool configuration.

Once all the required relationships have been added, the Service Pool can be considered complete (See Figure 11).

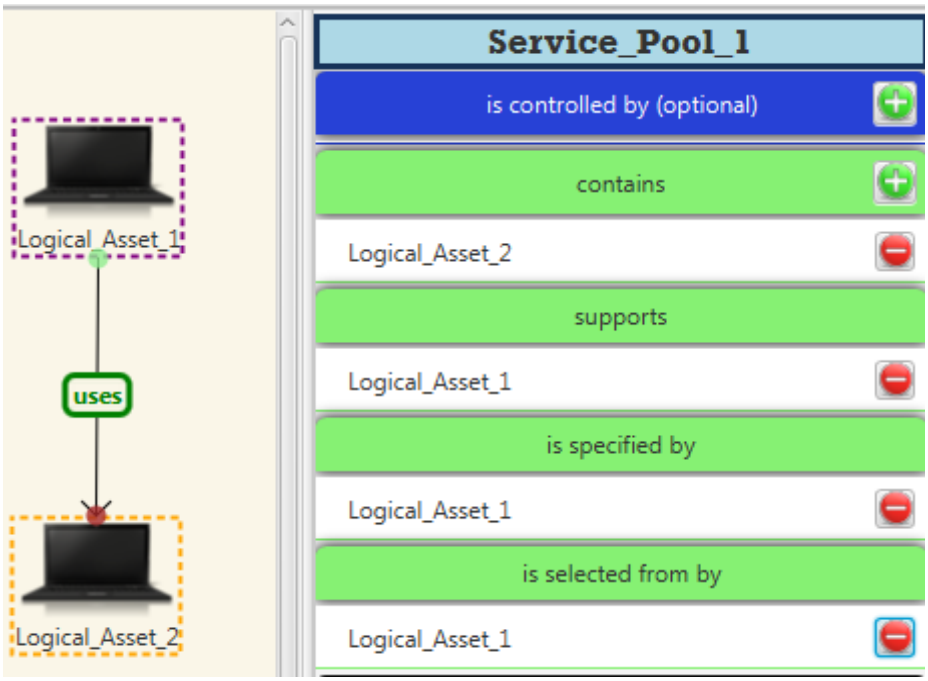


Figure 10: Completed Service Pool

TWME supports all the basic functions, such as rearranging, renaming and deleting assets. Usability of the models has been taken as prime consideration. Factors for usability were determined after holding focus groups with various actors involved in the OPTET lifecycle such as system designers, security and semantic experts and end users of use cases.

Selecting an Asset (including selecting a ServicePool by selecting the uses relationship) will show information about this Asset, about incoming and outgoing relationships, super classes and threats (if any have been generated for the Asset): (Figure 12)

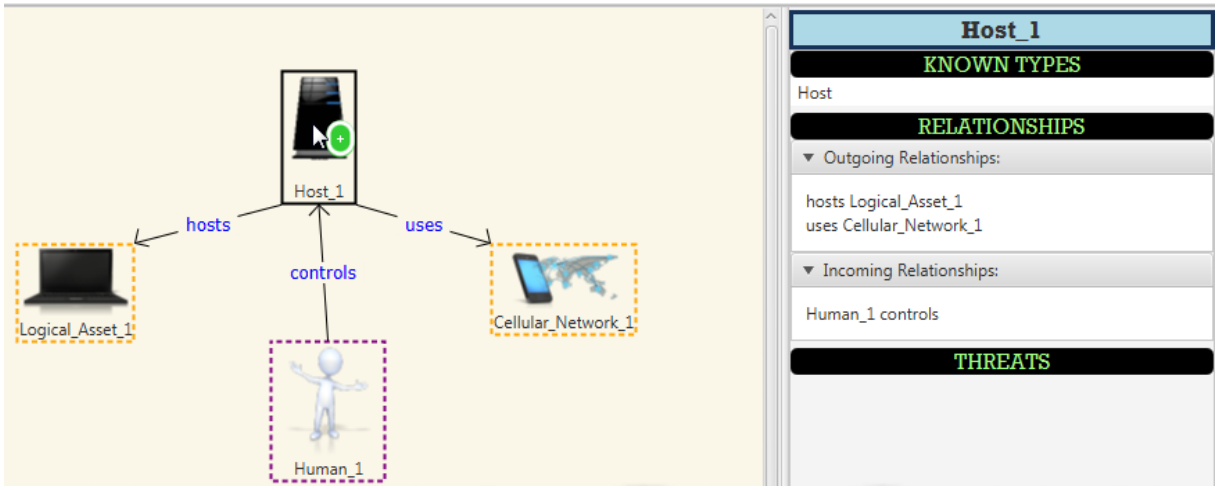


Figure 11: Asset information pane.

The key function of the Trustworthiness Model Editor is to display the possible threats to an Abstract System Model that has been designed. In order to do this, there are two buttons, one to create a local model and one to compile (Figure 13).

Figure 12: Save and compile options.

The "Create" button creates an OWL ontology file of just the Abstract System Model structure that has been designed. This model will be saved in a file "*x*_created_model.owl" in the current working directory, where x is the name that you have given your system model. If there is no current working directory, you will be prompted to select a save location for the ontology model.

The "Compile" button creates an ontology model of the Abstract System Model structure that has been designed and saves it, in the same way as the create button does. The local model is then merged with the Generic Model and inferences are run on this, in order to deduce the different subclasses each Asset in the system belongs to and any threats that would be present on each of the assets. The System Compiler component which is internal to the TWME is invoked first in order to determine all the possible class types for the assets in the created model.

1.1.2. System Model Compiler (SMC)

The System Model Compiler performs 2 different functions:

Firstly, it creates assert class types which are missing but are necessary in order to perform a full threat analysis on a given system. (For example packet flooding attacks are common on interfaces between Network and Host assets. The Interface class needs to be automatically created). The reason the Interface classes are not shown in the GUI is to avoid the introduction of a large number of components on screen which may confuse the user. Also, the rule for generation of Interface classes does not require human intervention. Figure 14 shows the pattern for interface generation as explained in D2.2 [7] (see Figure 14).

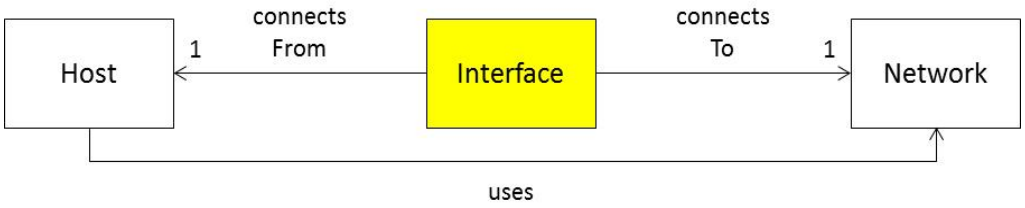


Figure 13: Interface class assertion pattern.

Any gained information about an asset will be added to the ontology and can be displayed by selecting the asset. This rule is encoded in the ontology using the SPARQL + SPIN [8] syntax and reasoning is performed on top using the TopSPIN semantic reasoner. Table 1 below shows the pseudo code for rule generation (The SPIN rule code itself is very large and can be viewed directly from the ontology which is available in the OPTET SVN)

```
Find all hosts that use or include a network
FOREACH of the found pairs DO
        Only consider only host/network combination that don't share a common interface yet
        Create a new interface to connect host and network
```

```
        Create the necessary restrictions to connect the newly created interface to the
        related assets

DONE
```

Table 1: SPIN pseudo code to asset Interface classes.

There are similar rules for the generation of Internal and External Network Group classes.

The second function of the SMC is to classify the asset types on the TWME canvas into specific types based on their interaction with other assets within the system. This determines the role of the asset. (e.g. Client, Service, Consumer, Delegate, Agent, Application Operator, Host Operator etc). The details of all possible types of patterns were described in D2.2. Figure 15 shows the pattern necessary to deduce consumer classes:

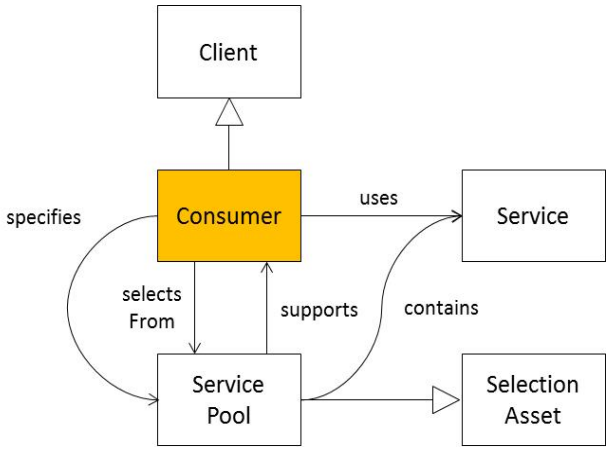


Figure 14: Consumer class interaction pattern.

The corresponding SPARQL + SPIN rule (pseudo code) for the above pattern is show in the Table 2.

```
IF Assets {Client, Service, ServicePool} exist

AND Relations {Client uses Service, Client specifies ServicePool, Client selectsFrom
SericePool, ServicePool supports Client, ServicePool contains Service} exist

THEN Client is a Consumer
```

Table 2: Pseudo code for Consumer class interaction pattern.

There are several options to be set before compiling a model. This gives the system designer more control over what shall be included in the compiled design-time model. Also there is the possibility to specify an ontology directory which enables the system designer to keep separate versions of the generic model and offers room for further customisation.

Once the SMC component has finished its process, the System Analyser is invoked.

1.1.3. System Analyser

The goal of the automated threat generation is to infer which generic threats apply to the assets within the current system. This is done by scanning the system asset model for threatened patterns defined in the generic model. In the ontology terms, this corrersponds

to creating system specific subclasses of generic threats. Figure 16 shows the ontological OWL representation of a threat S-14.07-ClientImpersonation as being a subclass of ServiceThreat, has a specific human readable description, *involves* the Service and Client assets, and threatens the Service asset. You will also notice that it has a SPIN rule called S-14.07-ClientImpersonation-MitigatedRule attached to it, which contains a control rule encoded in SPIN + SPARQL syntax which defines how this particular threat can be mitigated.

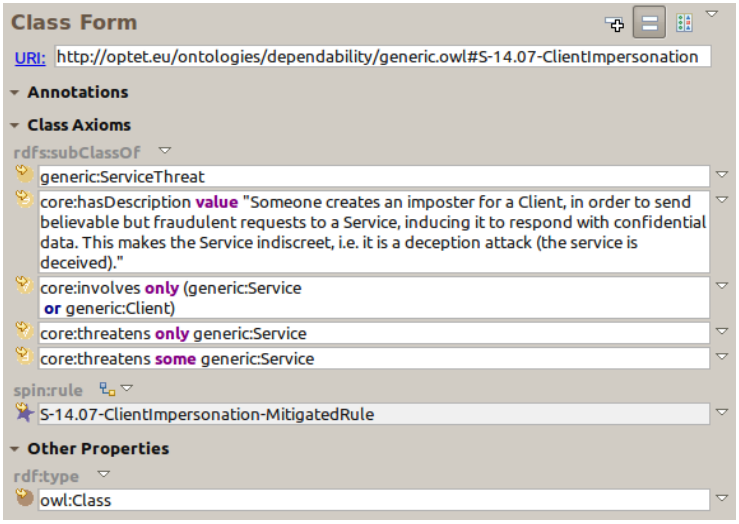


Figure 15 - Example of a generic threat class

The threat shown in Figure 16 applies to the Client-Service pattern from the generic model. It threatens the Service in this pattern and has a mitigation rule.

Every generic threat class also has a corresponding threat generation template encoded in SPIN which is part of the generic model. All rules are based on the same algorithm:

- Find a pattern in the system specific design-time model
- Create a new system-specific subclass of the generic threat
- Apply new system-specific restrictions to the newly created class

The threat generation rule works in the following way:

The pattern described in a WHERE clause (e.g. Client uses Service) is searched for in the system specific design-time model. This rule works at a Class (OWL class) level and does not automatically classify instances at run-time. It has been encoded this way on purpose as this is a design-time modelling process.

A BIND command generates the name for a new, system-specific subclass of the generic threat, which includes the names of the involved assets.

A CONSTRUCT clause then creates a new owl:Class with this name and adds the necessary restrictions to it. If this rule is for example run on the AAL design-time model, it will encounter the LogicalAsset PERSApplication which uses AdviceDispatchingService. According to the rule, it will create a new system-specific subclass of S-14.07-ClientImpersonation called S-14.07-ClientImpersonation-PERSApplication- AdviceDispatchingService that has as restrictions "threaten AdviceDispatchingService" and "involve both AdviceDispatchingService and PERSApplication".

Once the threat generation has been completed by the System Analyser GE, the results are sent back to the TWME GUI to be displayed to the user whenever they click on any asset on the canvas. Figure 17 shows the panel where these outputs are shown.

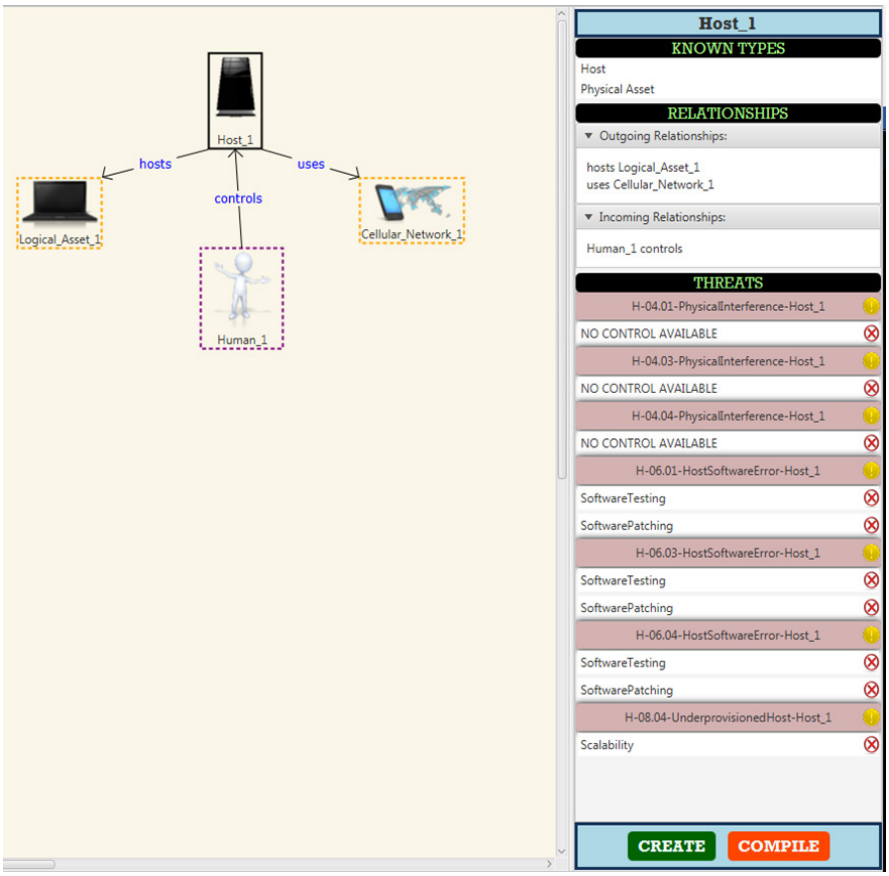


Figure 16 Threat information displayed to the user

Having generated a list of threats to each asset in the system model, it is possible to find out and distinguish more information about each of them. Double click on a threat in an Asset's information panel in order to bring up its corresponding dialog box as shown below:

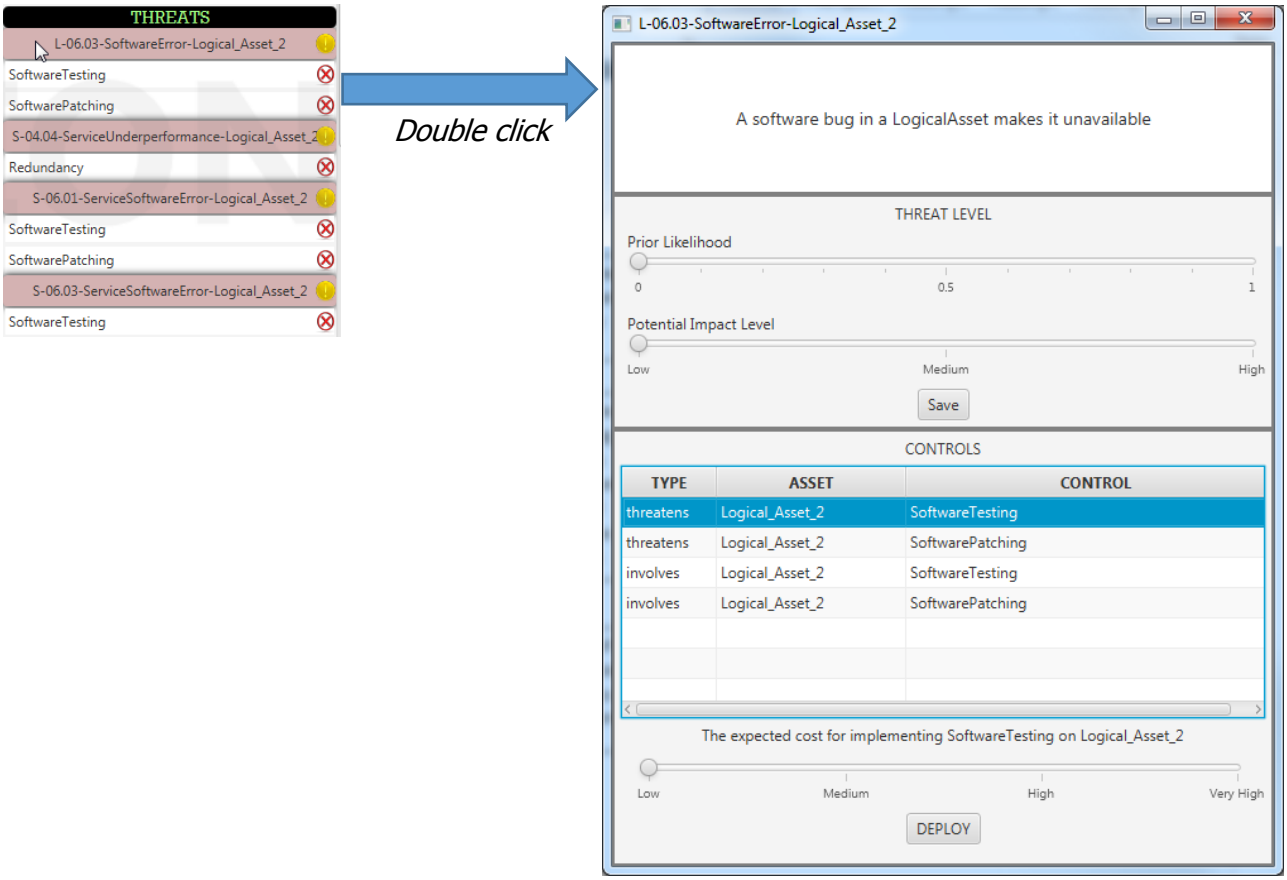


Figure 17 Threat Details

In the future, the TWME GE will add support to set the prior likelihood of the threat occurring and the level of impact that this threat would have on the system (this is currently set manually using an existing ontology editor e..g Protege, Topbraid). This will provide basis for threats classification and prioritisation. Moreover, more information will be provided about the control objectives. The controls table displays the list of control objectives to counteract this specific threat. Each control objective can be implemented using a set of controls. The controls can be further qualified for instance as deterrant, preventative or corrective and their cost estimated for further threat prioritisation.

We intend to make it possible to export this additional information as reports to answer the questions different stakeholders might have ("How much will it cost to make the system secure?", "What threat will be the most expensive to mitigate?", "What assets put the whole system at risk?")