# Ontology-based Business Knowledge for Simulating Threats to Corporate Assets

Andreas Ekelhart, Stefan Fenz, Markus D. Klemen, A Min Tjoa, and Edgar R. Weippl

Secure Business Austria – Security Research, A-1040 Vienna, Austria
{aekelhart, sfenz, mklemen, atjoa, eweippl}@securityresearch.at
WWW home page: `http://www.securityresearch.at`

**Abstract.** We propose a security ontology, to provide a solid base for an applicable and holistic IT-Security approach for SMEs, enabling low-cost threat analysis. Based on the taxonomy of computer security and dependability by Landwehr [ALRL04] and the threat classification according to Peltier [Pel01], a heavy-weight ontology can be used to organize and systematically structure knowledge on threats, safeguards, and assets. The ontology is used in an organization to capture business knowledge required for and created during a security risk analysis where instances of concepts are added to the ontology to allow the simulation of different attack and disaster scenarios. Each scenario can be replayed with a different protection profile as to evaluate the effectiveness and the cost/benefit ratio of individual safeguards.

## 1 Introduction

IT-Security is no longer limited to access control or preventing the classical virus attack; applied IT-Security also has to consider social engineering, acts of nature beyond human control, industrial espionage or physical attacks. With the need to implement IT-Security measures in almost every environment and faced with the growing application scope, it becomes increasingly difficult for experts of different domains to understand each other and to use a precisely defined terminology.

Nowadays, a well working IT-infrastructure is business critical for almost every enterprise and those who take IT-Security seriously spend a lot of money on maintaining and securing this infrastructure.

Compared to large companies, Small and Medium Sized Enterprises (SMEs) are usually not financially able to employ an own IT-division to plan, implement and monitor a holistic IT-Security concept. In a lot of cases only one or two employees are responsible for maintaining and securing the entire IT-infrastructure. Mostly, this circumstance leads to overwhelmed IT-administrators who are not able to guarantee the implementation of a holistic IT-Security concept. [Hau00] summarized the problems of SMEs regarding the IT-Security aspect:

- Smaller IT budget, relative to total budget as well as in absolute figures

- Less IT knowledge, information technology is often looked after by employees from other departments
- IT is not considered as important as within larger enterprises although more and more core processes are processed by IT elements
- IT environments are not homogeneous

In many cases these points cause a very poor implemented IT-Security concept and a security ontology can help in a first step, to clarify the meaning and interdependence of IT-Security relevant terms [Don03]. Beside the term definition we have also to integrate the dependability of threats, countermeasures and resources. The integration of these dependabilities is necessary to model events which threaten existing resources and for each threat we have to model proper countermeasures which are able to secure the resources. Such an ontology combined with an user interface would enable SMEs, to run their risk management and threat analysis for low costs and without an expensive audit program like CobiT [COB06] or ISO17799 [ISO06].

## 2  Security Ontology

The entire security ontology consists of three parts: The first part is derived from the security and dependability taxonomy by Landwehr [ALRL04], the second part describes concepts of the (IT) infrastructure domain and the third part provides enterprises the option to map their persons and internal role models. The taxonomy is designed in a very general way, and so it may easily be extended with additional concepts.

The ontology is coded in OWL (Web Ontology Language [OWL04]) and the Protege Ontology Editor [Pro05] was used to edit and visualize the ontology and its corresponding instances. The following subsections describe the parts in more detail:

**Attribute Ontology**  Each instance in the *Threat* ontology impacts $n$ security attributes but the actual impact is not expressed in numbers. In fact the ontology shows which threats influence certain security attributes which is useful if a company wants to prioritize the IT-Security strategy regarding specific attributes.

**Threat Ontology**  Figure 1 shows an excerpt of the most important parts of the *Threat* ontology and the relevant relations to the other ontologies. As already described above we derived the fundamental structure from [ALRL04] and [Pel01] respectively and extended it with own concepts and relations. The *Threat* ontology with its various relations, represents a central part of the entire security ontology and makes the mapping of threats including proper countermeasures, threatened infrastructure and proper evaluation methods possible. The most important relations in a nutshell: Any instance of concept *sec:Threat* or one of its sub-concepts affects $n$ instances of concept *Attribute* (e.g. Availability, Integrity,

Maintainability) and with *sec:preventedBy* and its inverse relation it is possible to map mitigating countermeasures to a certain threat. To enable the mapping of threatened infrastructure to a defined threat the relation *sec:threatens* was introduced, where each threat threatens $n$ infrastructure elements. Last but not least the *sec:evaluatedBy* relation enables the user to map certain evaluation methods (e.g. probabilistic methods, qualitative methods, quantitative methods) to a defined threat class to support the user and the ontology-consuming application respectively at the risk analysis. Figure 2 shows a practical implementation of
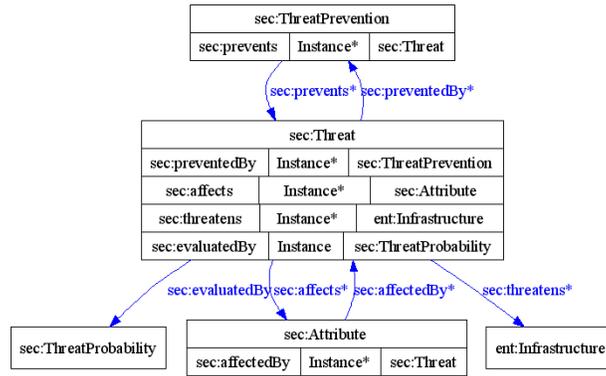


**Fig. 1.** Threat ontology - concept relations

the computer virus instance *Tequila* and provides the reader with real world values regarding the computer virus concept. It can be seen that each network and computer device is described by its physical (in our example the development server and the switch are located in room R0202) and virtual (location within the network ⇒ development server is connected to the switch) location. If there would be no anti virus program installed on the development server and the affected operating system would be equal to the installed operating system the ontology reasoner would connect with the *sec:threatens* relation the instances *Tequila* and *DevelopmentServer* to indicate a potential threat. *sec:affects* and its inverse relation maps a certain threat to the corresponding security attributes and vice versa. In our example availability, reliability and integrity are directly affected by the computer virus instance *Tequila* and it has to be noted that the computer virus threat acts only as an example for further threats which can be modeled in a similar way.

**Means Ontology** Figure 3 shows an excerpt of the *Means* ontology and we can see that relation *sec: requires* connects the concepts *sec:TechnicalThreatPrevention* and *sec:ThreatPrevention* to indicate that some technical threat prevention instances are requiring instances of the *sec: ThreatPrevention* concept. The security ontology needs this relation to determine countermeasures which consists of
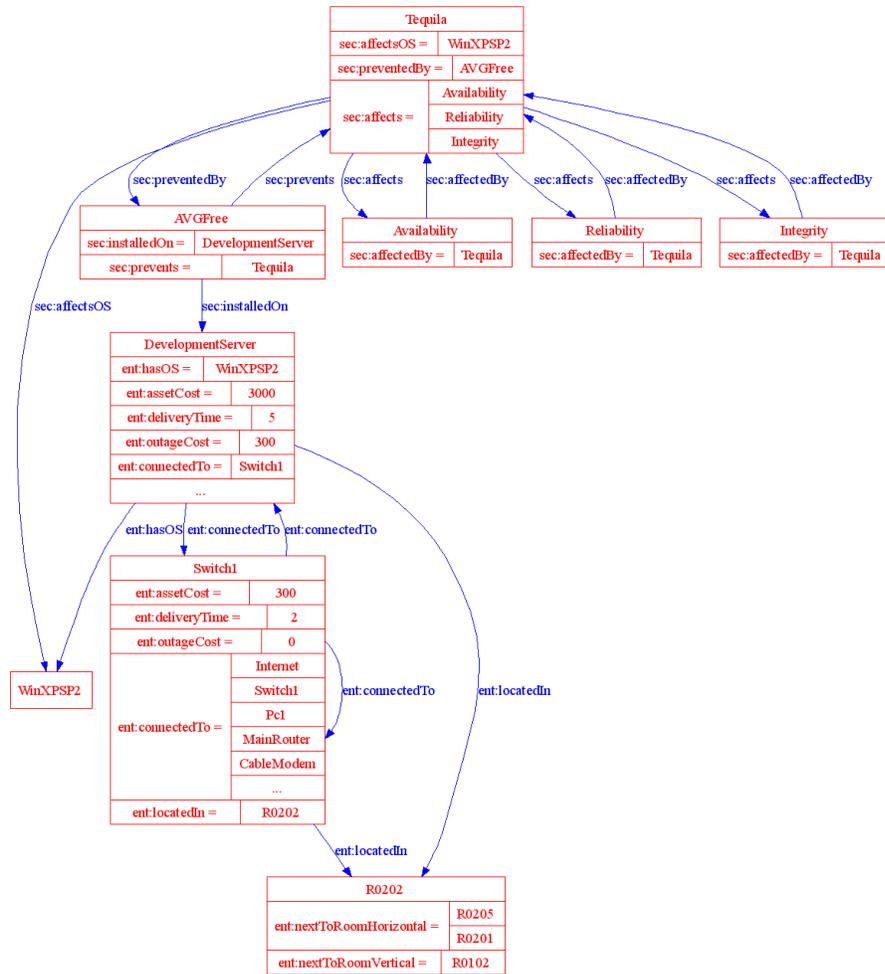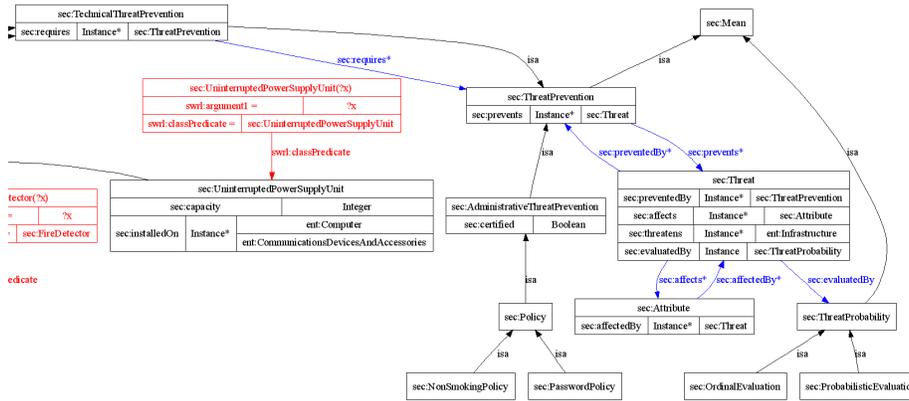
**Fig. 2.** Computer virus instance *Tequila* and corresponding relations

various elements such as a fire extinguisher system and a smoke detector were both elements are necessary to ensure an effective countermeasure.



**Fig. 3.** Means ontology - concept relations

**Infrastructure Ontology** Figure 4 shows a fragment of the infrastructure part of the security ontology and due to the size of the infrastructure ontology it is not possible to show the entire visualization and so we extracted the most relevant part. The company building with its corresponding floors and rooms can be described by using the infrastructure framework and to map the entire building plan exactly on the security ontology, each room is described by its position within the building. The ontology is able to reconstruct the entire building virtually by the following relations: (1) *ent:hasFloor* describes which instances of concept *Floor* can be found in a certain building (2) *ent:isUnderFloor* was designed to determine those floors which are located under a certain floor (3) *ent:isAboveFloor* defines the same type of knowledge as (2) but in the inverse direction (4) *ent:nextToRoomHorizontal* and *ent:nextToRoomVertical* define those rooms which are horizontal/vertical next to a certain room.

**Role Ontology** The integration of roles is necessary to map the correct hierarchies to the ontology. The current role taxonomy includes the *ent:administers* attribute that is used to specify for which instances of *ent:ComponentForInformationTechnology* an administrator (instance of *ent:Administrator*) is responsible for. Additional relations depending on the further usage are thinkable and can be implemented without any effort if needed.

**Person Ontology** This sub-ontology represents a simple listing of natural persons who are relevant for the modeling of certain security issues. Every person
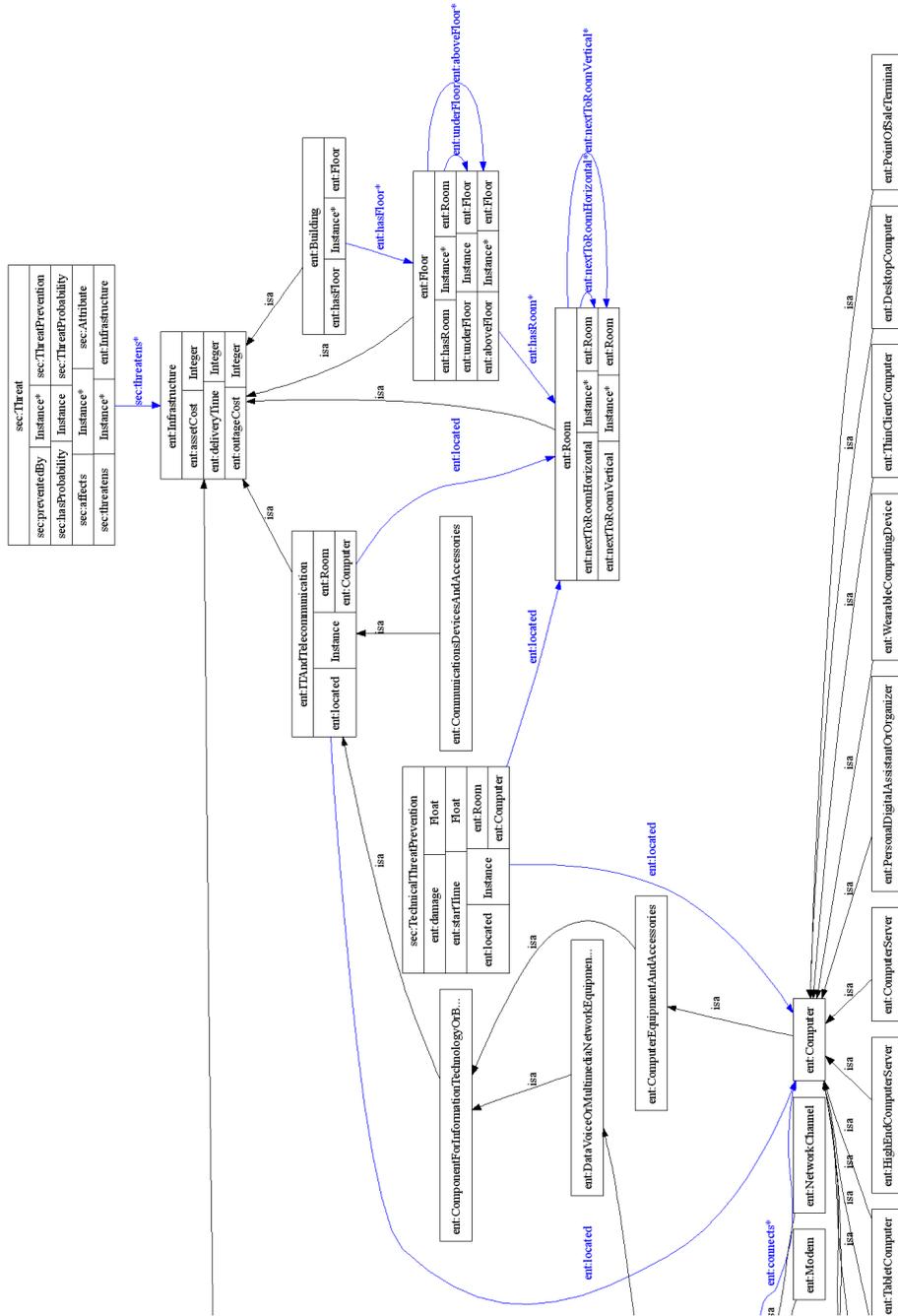
**Fig. 4.** Infrastructure ontology - concept relations

holds *n* roles and Figure 5 shows the most important concept relations of the *Person* taxonomy: the *ent:usedBy* relation maps computers, personal communication devices, etc. to its regular users and enables the ontology to relate those concepts for a further threat simulation (e.g. the calculation of non-productive personnel costs in the case of broken computers and/or communication devices).



**Fig. 5.** Person ontology - concept relations

### 2.1 Ontology maintenance

The goal of this approach is to make detailed threat analysis possible, without requiring a lot of experts and time. Experts are only needed for the framework creation (at the first two levels). Three levels exist:

1. Defining the concepts of the ontology (including infrastructure, persons, roles and disasters concepts), must be obviously done by experts.
2. On basis of these concepts domain experts have to provide individuals (e.g. disasters, safeguards) and their attributes (e.g. spread time, damage, probability of occurrence).
3. At this stage a solid framework exists which can be utilized by companies. It is their task to model their company and this process will be done by the

administration or in further versions automatically based on building plans and the information extracted by automatic IT infrastructure recognition tools. Slight customization and extension can be done by company intern staff or extern experts.

Filling the ontology is supported by user friendly and intuitive forms. The application itself, which accesses the ontology knowledge and executes the simulation to calculate risk management ratios, is provided by us. The next section presents example queries against the knowledge base.

## 2.2 Querying the ontology

To take advantage of the developed ontology we must receive consistent answers as return on questions against the knowledge base. In the following we provide two easy examples, written in SPARQL (an RDF query language)[SPA06].

'What technical threats affect Integrity?', could be such a question from an administrator who wants to get to know unnoticed threats to check and update his security policy. The SPARQL representation is as follows:

```
SELECT ?threat
 WHERE {  ?threatClass  rdfs:subClassOf  sec:TechnicalThreat  .
          ?threat  rdf:type  ?threatClass  .
          ?threat  sec:affects  <#Integrity> }
```

The query returns every instance of subclasses of *sec:TechnicalThreat* (see Subsection 2), e.g. 'Win32.Blackmal.F', 'Fire', etc. Now when the administrator knows possible threats he could ask for threatened infrastructure elements in his business, concerning the worm 'Win32.Blackmal.F'. The query can be defined like this:

```
SELECT ?threatenedAsset
  WHERE {  <#Win32.Blackmal.F>  sec:threatens  ?threatenedAsset  }
```

As result he gets a list of Windows-based computers in his company: 'Pc1, Pc2, FileServer, ...'. Obviously these are easy but already effective examples on how to use the developed ontology. Further queries could ask for possible countermeasures or the status of the anti-virus instances running on the PCs. In the following chapter we will introduce an advanced example on how to benefit on this approach.

## 3    Scenario

In this section we provide an example of how a company would use the aforementioned security ontology to model an IT infrastructure.

## 3.1  The Company

The company is an SME with six employees. Their main business is software sales and custom programming to modify their standard software. The company rents two floors (1st and 2nd floor) of a 5-floor building in the center of a small town. The following listing shows the allocation of relevant (IT) infrastructure elements:

- First floor - Office room (R0103): 2 PC's
- First floor - Storage room (R0104): data media (archived)
- Second floor - Server room (R0202): 4 Servers, 1 Router, 2 Switches, 1 Modem
- Second floor - Office room (R0203): 1 PC, 1 Notebook
- Second floor - Office room (R0204): 3 PC's

The infrastructure is mapped on the sub-tree *ent:Infrastructure* (compare Figure 4) and the following example gives an idea regarding the attributes of all *ent:Infrastructure* elements:

The concept *ent:PersonalComputer* with its concrete instance 'Pc4' has the attributes *ent:deliveryTime*, *ent:assetCost*, *ent:outageCost* and *ent:located*. If this or any other instance will be destroyed by a certain disaster, the ontology 'knows' how long it takes to get a new one, how much it costs, where it is located and the outage costs per day.

## 3.2  The Disaster

After describing the company with its infrastructure, the current subsection defines the disaster, which will hit our software company.

The event of fire, as a physical threat scenario, was chosen. The simulation should show the amount of damage in the course of time and in consideration of the fire source. A certain room can be defined as the fire source; the speed of propagation without any countermeasures will be 5 minutes per floor and 5 minutes per room. Every infrastructure element is assigned to a certain room. In the case of fire all infrastructure elements within a room will be destroyed completely. The outage costs per room correspond to the outage costs sum of all destroyed elements, which are located in the room. It is possible to assign countermeasures to any room. These safeguards can lower the probability of occurrence and the speed of propagation in the case of fire. The attribute *ent:damage* addresses the damage which results when the countermeasure is executed.

For instance: *ent:WaterFireExtinguisher* is located in room R0102 and will start, when switched on, immediately. Instance 'WaterFireExtinguisher0102' will extinguish the room within one minute. The attribute *ent:startTime* is important for countermeasures which are not activated automatically (e.g. hand fire extinguisher).

### 3.3   The Simulation

The framework for our threat analysis has been explained in the preceding sections, now we present a tool called *SecOntManger* which processes the ontology knowledge to simulate threats. This prototype handles IT costs and poses as proof of concept. Further threat effects as well as infrastructure components can be added easily due to the generic structure.

In our example the management wants to know what impact fire would have on the infrastructure, what countermeasures exist and what their benefits are. For this purpose we show two program runs, one against the unprotected company another including safeguards.

– The first program run without countermeasures: *SecOntManager* offers an intuitive graphical user interface, shown in Figure 6. A threat and a corresponding starting point have to be chosen before a simulation can be started. We decide for fire as threat and the server room (Room0202) as origin of fire. The program run produces a detailed log file which shows how the fire spreads from room to room and what damage it causes.
   Each room is processed completely before neighboring rooms are searched. At the end of the simulation all occurring costs are visualized in a line chart (see Figure 6). The time axis unit is set to minutes. Four curves, reflecting different cost categories, exist. The blue curve visualizes the damage: In the example the damage costs rise very fast due to the speed of fire - within 30 minutes every room was destroyed. By zooming in, displaying only the first 30 minutes, we can see how the damage evolves. After every room 'burned down' no further damage can occur. Red are the outage costs, taken from assigned outage costs of infrastructure components and employee's costs. Outage costs rise constantly in the simulation until recovery. The green curve shows recovery costs; delivery times for destroyed components are taken into consideration. When components are available and paid, connected outage costs decrease, visually spoken, the red line flattens. Additional installation costs lift the recovery costs upon damage. When every component is recovered, the pre-threat state is reached and outage costs do not rise anymore. Furthermore the total of all costs is reflected by the yellow curve. Fire costs 73605 Euros and it takes at minimum five days to recover the IT-infrastructure from the effects.
– Second program run with countermeasures enabled: We now concentrate on reducing the damage by installing safeguards. *SecOntManager* offers to install fire suppression systems in the building. We decide for pre-action pipes in the entire building. Necessary detectors and fire extinguishers are added to rooms in the OWL file. Their costs amount to 7200 Euros. Running the simulation produces the cost chart in Figure 7. As can be seen the total damage decreased drastically to 28329 Euros. After installing safeguards, the fire can not spread anymore; it is detected and extinguished shortly after breakout. Nevertheless costs and recovery times are still very high. The reason is that water extinguishers have a high damage factor concerning electronic devices
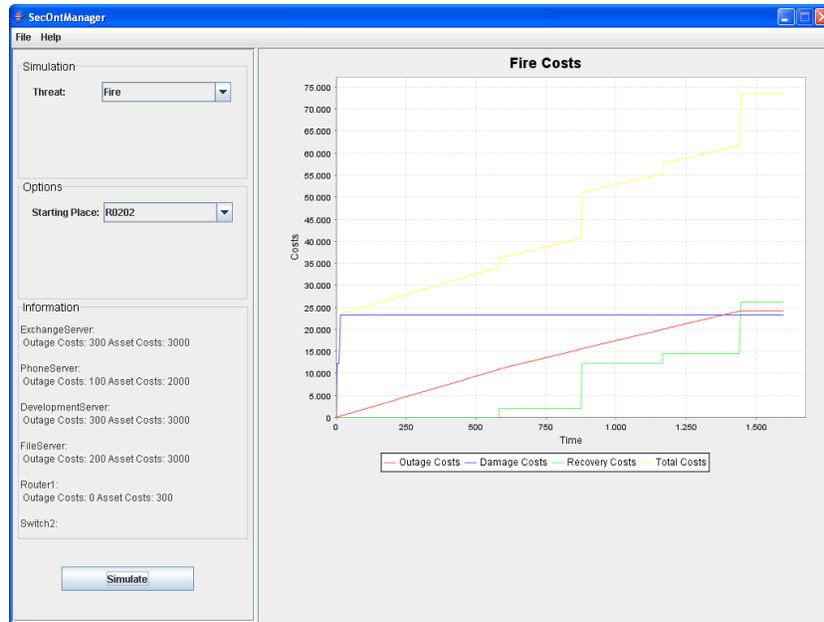
**Fig. 6.** SecOntManager: No countermeasures

and we have chosen the server room as place of fire origin. *SecOntManager* also offers CO2 fire extinguishers for locations with high electronically damages. Replacing the water extinguisher by a more expensive C02 extinguisher the total costs are reduced to 10934 Euros, which are mostly outage costs of one server which caused the fire. By adding a redundant server the outage time and costs could be cut to zero.

## 4   Conclusion

We propose an ontology-based approach combining security- with business-domain knowledge to model companies. The ontology guarantees a shared and accurate terminology as well as portability. Knowledge of threats and corresponding countermeasures, derived from IT-security standards, is integrated into the ontology framework. Moreover, we implemented a prototype capable of simulating threats against the modeled company by processing the knowledge contained in the ontology. *SecOntManager* visualizes the damage caused by specific threats, outage costs and the recovery time. Running the program with added safeguards shows their benefits and offers objective data for decision making, which safeguards to implement and to avoid installing countermeasures that are not cost-effective. An enhanced prototype with advanced risk analysis and broader threat support will be developed in further research activities.
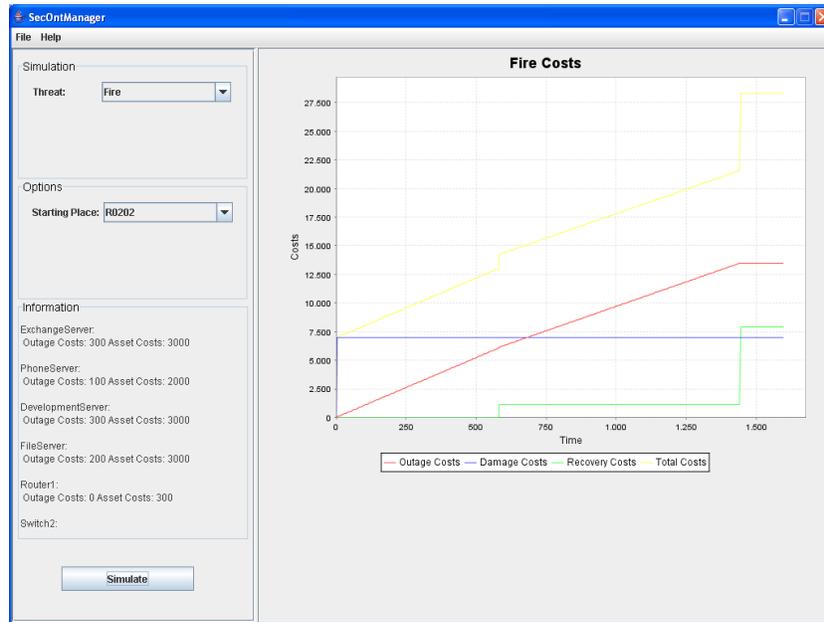
**Fig. 7.** SecOntManager: With countermeasures enabled

# Acknowledgements

# References

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, 1(1):11–33, 2004.

[COB06] Cobit. http://www.isaca.org/, 2006.

[Don03] Marc Donner. Toward a security ontology. *IEEE Security and Privacy*, 1(3):6–7, May/June 2003.

[Hau00] Hans Eduard Hauser. Smes in germany, facts and figures 2000. Institut für Mittelstandsforschung, Bonn, 2000.

[ISO06] Iso17799. http://www.iso.org/, 2006.

[OWL04] Owl web ontology language. http://www.w3.org/TR/owl-features/, 2004.

[Pel01] T. R. Peltier. *Information Security Risk Analysis Boca Raton*. Auerbach Publications, Boca Raton, Florida, 2001.

[Pro05] The protege ontology editor and knowledge acquisition system. http://protege.stanford.edu/, 2005.

[SPA06] Sparql query language for rdf. http://www.w3.org/TR/rdf-sparql-query/, 2006.