



Semantic Web Implementation Methodology (SWIM)

SCOTT STREIT, INTERWISE CHIEF TECHNOLOGY OFFICER

SWIM has been endorsed by the Massachusetts Institute of Technology Research Establishment (MITRE) and the National Institute of Standards and Technology (NIST).

Step 1

Describe your initial, most difficult requirements in conversational, informal English. Leverage any existing diagrams or formalisms.

Step 2

Decompose the problem into domain components. Pick the most difficult domain as a starting point.

Step 3

Look for opportunities of abstraction to lessen the number of components.

Step 4

Research existing vocabularies and ontologies in similar domains to use in composition.

Step 5

If a preexisting vocabulary does not exist, model it yourself creating Tbox entries.

Step 6

Take an instantiation of the data and prove it can work on paper.

Step 7

Use a semantic web implementation, like Jena, to build a Tbox vocabulary.

Step 8

In a semantic web implementation, instantiate the vocabulary by creating instances and output the instances as RDF/XML.

Step 9

Iterations - repeat steps 1 through 8 until complete.

National Institute of Standards Technology (NIST) Case Study

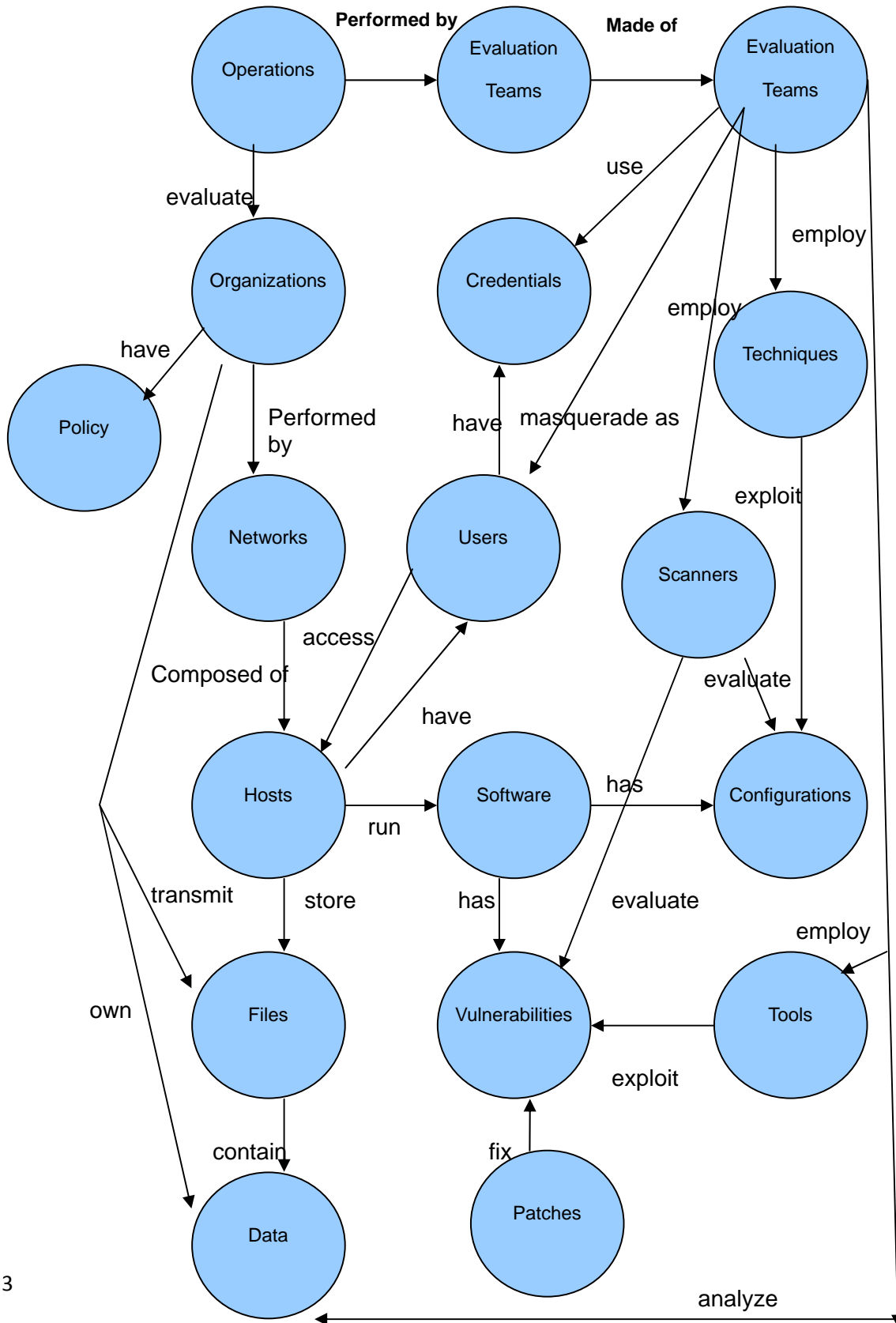
For this project, NIST agreed to be the domain holder for PRODUCT, TOPIC, VULNERABILITY, PRODUCT, PRODUCT-INSTANCE, and CONFIGURATION. As a result, they are providing web server setup that provides documentation for the namespace and URI. Additionally, NIST will manage versions and releases of the domain vocabularies.

Step 1 - Describe your initial, most difficult requirements in conversational, informal English

For naming IT platforms subject to vulnerability and configuration guidance, patching and remediation, asset management, and other security related tasks, there are three distinct parts of a platform that need to be addressed:

1. Hardware
Hardware is the physical platform supporting the IT system. The type and model of hardware can be relevant for some guidance and vulnerabilities.
 2. Operating System
The operating system controls and manages the IT hardware and supports applications. The operating system type, version, edition, and upgrade status are almost always relevant for vulnerability descriptions and guidance.
 3. Application Environment
Software systems, servers, and packages installed on the system are often relevant for vulnerability and guidance. The diversity of applications that may be installed on a modern IT platform is very great, but typically a specific piece of guidance or a specific vulnerability description depends on only one or two applications.
- The system **MUST** be able to express each type of platform part described above.
 - The system **MUST** be able to express platform information across a wide range of specificity.
 - The system specification **SHALL** focus on enumerating platform types.
 - The system **MUST** be able to include the language a particular platform supports.
 - The system **MUST** define some means to specify concrete platform identification.
 - The system **MUST** exhibit the prefix property.
 - The system **MUST** enforce the creation of unique terms for a given name.

Figure 1 – Description and Relations of business entities



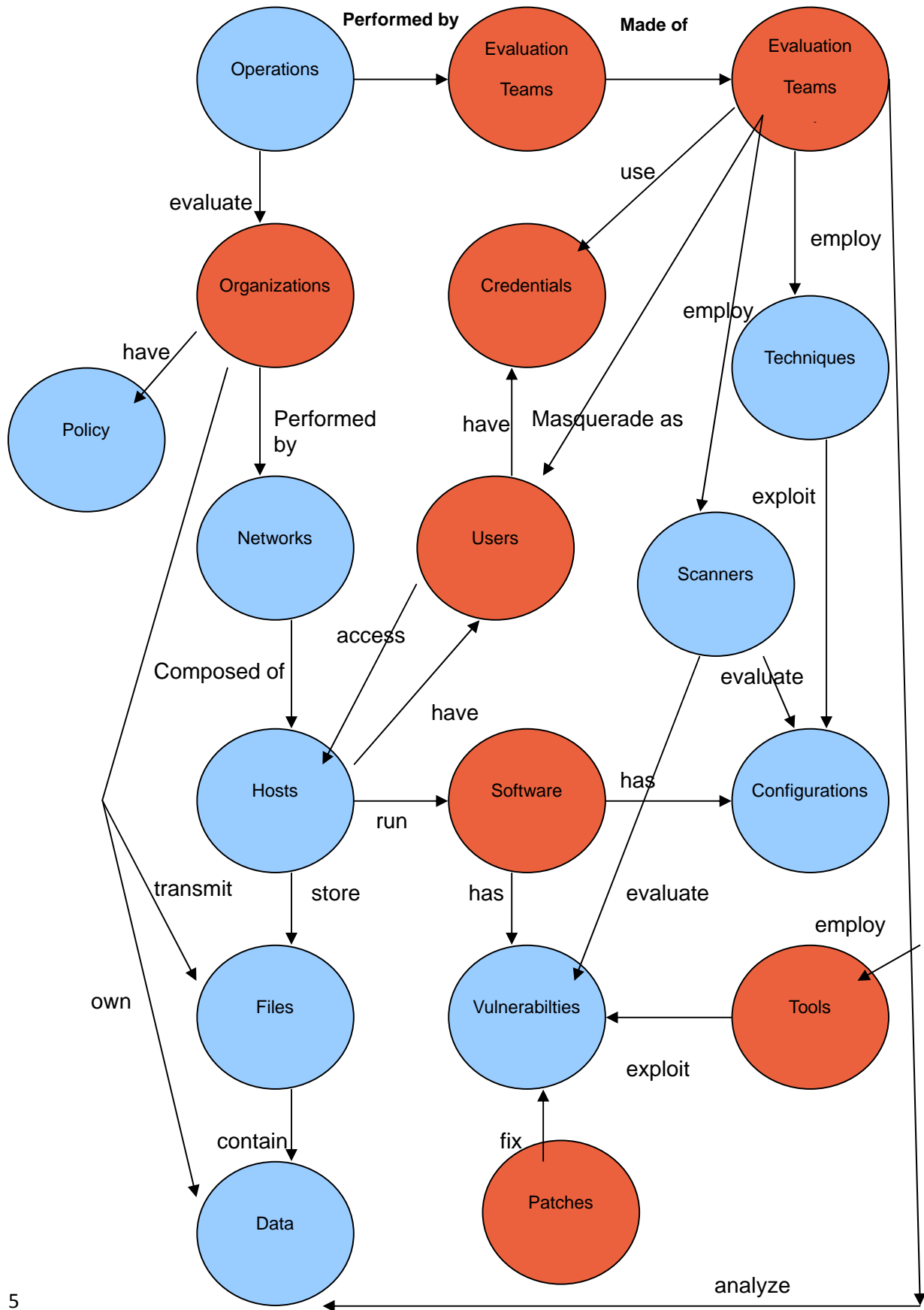
Initially, the focus is on Users, Evaluation Teams, Evaluation Team Members, Credentials, Organizations, Configurations, Software, Tools and Patches.

Step 2 - Decompose the problem into domain components. Pick the most difficult domain as a starting point

Clearly, the software/tools domain is the most complex. We add in Users, Evaluation Teams, Evaluation Team Members and Credentials since we know there are predefined Tbox for this domain and it gives us some proximity.

Because it is imperative to aim to describe the task at a high level of generality, it is not necessary to capture all nouns at the beginning. The group brainstormed and determined the highest level nouns, identifying Product Entity as an 'exploding' noun because products consist of products and products depend on other products.

Figure 2 – Areas of Focus (red)



Step 3 - Look for opportunities of abstraction to lessen the number of components

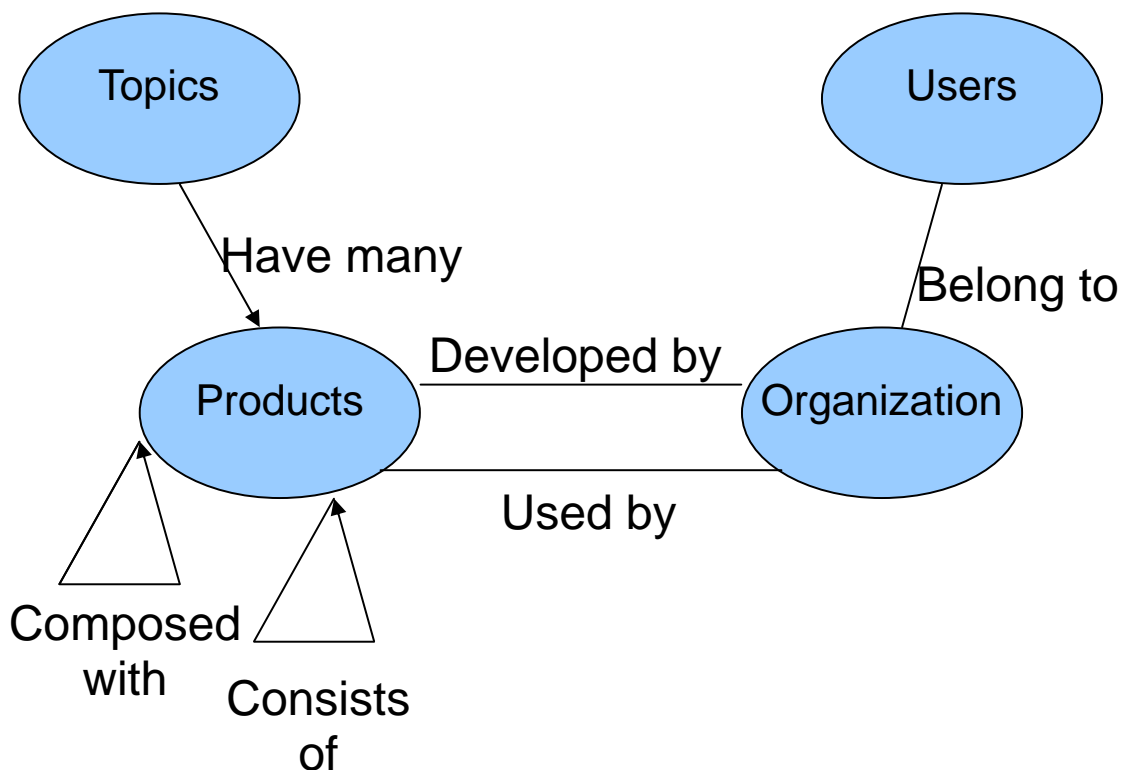
This Step is the “special sauce.” We see that Configurations, Software, Tools and Patches are really groupings of Products and Topics. If we use composition and inheritance we get an abstraction and a simplification.

Object composition (black-box reuse) is an alternative to class inheritance (white-box reuse). New functionality is obtained by assembling or composing objects to get more complex functionality. Object composition is defined dynamically at run-time through objects acquiring references to other objects. Any object can be replaced at run-time by another as long as it has the same type.

Favoring object composition over class inheritance helps keep each class encapsulated and focused on one task. The rationale is as follows:

- Classes and class hierarchies remain small and manageable.
- A design based on object composition has more objects (instances) and less classes (Tbox) and the system behavior depends on their interrelationships instead of being defined in one class.
- As stated by Erich Gamma, “Favor object composition over class inheritance. In our experience is that designers overuse inheritance.”

Figure 3 – An abstraction over the aforementioned red labeled notes



While it is important, to know the Cardinality and composition of products up front, i.e. what a template may look like, it is impossible to know the specifics and the measure to which the consistencies and dependencies may change.

- Product
- Organization
- People
- Topics/Classification (architecture, os, web, app server)

Product: hardware, software or system component

- **Has name**
 - Products may be known by more than one name (e.g., marketing name = “Vista”, codename = “Longhorn”, abbreviation = jre, alternative branding)
 - Identify a preferred name
- **Has vendor**
 - vendor can be commercial entity, educational institution, open source, government agency, individual
- **Consists of other products**
 - e.g., Microsoft Office consists of Word, Excel, PowerPoint
- **Has sub-products**
 - e.g., SQL Server has Reporting Services
- **Has version**
 - ability to serialize versions
- **Depends on products**

Organization:

- **Has name**
 - Organization may be known by more than one name (e.g., formal, informal, acquisitions, mergers)
- **Releases products**

Person:

- **Has name**
 - Person may be known by more than one name
- **Belongs to organization**

Topics/Classifications (architecture, operating system, web server, app server)

We may come up with topics that can be, Windows (Home Professional), Host environment, Target environment or the release of a language such as Java 1.5 or Java 1.6.

Step 4 - Research existing vocabularies and ontologies in similar domains to use in composition

Friend Of A Friend (FOAF) and Virtual Card (VC) are considerations for Users and Organizations. We choose VC for better support of Organizations and the relationship to Users. No suitable domain vocabularies exist for the remainder of the problem so we need a custom vocabulary for Product and Topic.

Step 5 - If preexisting vocabulary does not exist, model it yourself, creating Tbox entries

Tbox can be described as a schema or definitions of concepts, while Abox consists of records or definitions of individuals/objects. In Computer Science an Abox is an “assertion component” or a fact associated with a terminological vocabulary within a knowledge base.

The terms Abox and Tbox are used to describe two different types of statements in ontologies. Tbox statements describe a system in terms of controlled vocabularies, for example, a set of classes and properties. Abox are Tbox-compliant statements about that vocabulary.

Tbox statements are sometimes associated with object-oriented classes and Abox statements associated with instances of those classes. Together Abox and Tbox statements make up a knowledge base.

First, we create a Topic Tbox and a Product Tbox; then we compose these vocabularies in the same way VC is constructed. Current vocabularies violate Gamma's approach to composition using a "kitchen sink" type of approach. Vocabularies compose with other vocabularies and should be agile.

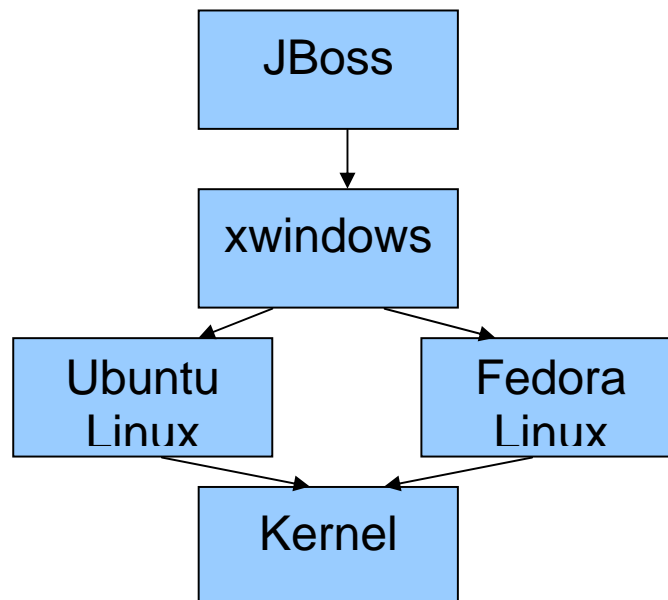
Topics allow grouping of items with information about a particular grouping. Any reference to an individual is thorough VC. Any reference to a product is through the Product vocabulary.

Step 6 - Take an instantiation of the data and prove it can work on paper

We track Linux as a product with a dependency on a Linux Kernel as well as xwindows. We place a JBoss application server on the stack and prove to ourselves that this paradigm will work.

We convince ourselves that we may compose products in a variety of different ways. This could mean anything from an embedded system using a Linux Kernel to a data processing system using JBoss. All may be modeled using our compositional product structure.

Figure 4



Step 7- Use a semantic web implementation, like Jena, to build a Tbox vocabulary

[Show source code for TOPIC.java and PRODUCT.java]

The <rdf:Bag> element is used to describe a list of values that is intended to be unordered. RDF defines several container elements. Collections

RDF provides a mechanism for creating collections, which are special kinds of Resources. A collection doesn't have to have its own URI, although it can. RDF supports three kinds of collections, bags, sequences, and alternatives, represented by RDF:Bag, RDF:Alt, and RDF:Seq elements respectively. A bag is just a bunch of Values, either simple strings or Resources. To create a bag, use an RDF:Bag element containing RDF:LI elements. Here is a simple bag of color names:

```
<RDF: RDF>
  <RDF: Bag>
    <RDF: LI >red</RDF: LI >
    <RDF: LI >green</RDF: LI >
    <RDF: LI >bl ue</RDF: LI >
  </RDF: Bag>
</RDF: RDF>
```

And here is a bag of Resources:

```
<RDF: RDF>
  <RDF: Bag>
    <RDF: LI resource="http://www.xml.com/xml/pub/98/06/rdf.html"/>
    <RDF: LI resource="http://www.cci.l.org/~cowan"/>
    <RDF: LI resource="http://www.w3.org/TR/WD-xml-names"/>
  </RDF: Bag>
</RDF: RDF>
```

We use a Bag to represent composition using other resources. A bag allows the resources used in composition to be unlimited and unordered.

Step 8 - In a semantic web implementation, instantiate the vocabulary by creating instances and output the instances as RDF/XML

Our code creates the RDF/XML (Tbox) entries based on our compositional vocabulary (Tbox).

```

package semantic.exp;

import com.hp.hpl.jena.rdf.model.Bag;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.DC;
import com.hp.hpl.jena.vocabulary.VCARD;

public class CPEModelGenerator {

    public static void main(String args[]) throws Exception {

        Model model = ModelFactory.createDefaultModel();

        model.setNsPrefix("product", PRODUCT.getURI());
        model.setNsPrefix("topic", TOPIC.getURI());
        model.setNsPrefix("organization", VCARD.getURI());
        model.setNsPrefix("action", ACTION.getURI());
        model.setNsPrefix("vuln", VULNERABILITY.getURI());
        model.setNsPrefix("score", SCORE.getURI());
        model.setNsPrefix("dc", DC.getURI());
        model.setNsPrefix("setting", SETTING.getURI());
        model.setNsPrefix("for", FOR.getURI());
        model.setNsPrefix("configuration", CONFIGURATION.getURI());

        Resource redHat = model.createResource("http://www.redhat.com/");
        redHat.addProperty(VCARD.NAME, "Red Hat");

```

```

Resource rhelAppServer = model
.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/1");
rhelAppServer.addProperty(PRODUCT.NAME, "cpe:/a:redhat:application_server");
rhelAppServer.addProperty(PRODUCT.TITLES, "Red Hat Application Server");
rhelAppServer.addProperty(PRODUCT.OWNER, redHat);
rhelAppServer.addProperty(PRODUCT.TOPIC, appServer);

Resource rhelKernel =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/2");

rhelKernel.addProperty(PRODUCT.NAME, "cpe:/a:redhat:kernel");
rhelKernel.addProperty(PRODUCT.TITLES, "Red Hat Kernel");
rhelKernel.addProperty(PRODUCT.OWNER, redHat); // is-a vendor
rhelKernel.addProperty(PRODUCT.TOPIC, kernel);

Resource rhel = model
.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/3");
rhel.addProperty(PRODUCT.NAME, "cpe:/o:redhat:enterprise_linux");
rhel.addProperty(PRODUCT.OWNER, redHat);
rhel.addProperty(PRODUCT.VERSION, "enterprise_linux");

Bag titles = model.createBag();
titles.add("Red Hat Enterprise Linux");

rhel.addProperty(PRODUCT.TITLES, titles);

rhel.addProperty(PRODUCT.TOPIC, os);

Bag container = model.createBag();

```

```

Resource appServer = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/1");
    appServer.addProperty(TOPIC.NAME, "Application Server");

Resource kernel = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/2");
    kernel.addProperty(TOPIC.NAME, "Kernel");

Resource os = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/3");
    os.addProperty(TOPIC.NAME, "OS");

```

```

container.add(rhelKernel);

rhel.addProperty(PRODUCT.CONTAINS, container);

appServer.addProperty(TOPIC.PRODUCTS, rhelAppServer);

kernel.addProperty(TOPIC.PRODUCTS, rhelKernel);

```

```

    // output...
    model.write(System.out);
    model.write(System.out, "N-TRIPLE");

}
}

```

Resulting Output

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:setting="http://nvd.nist.gov/ontology/SETTING/0.92#"

```

```

  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <rdf:_1>Red Hat Enterprise Linux</rdf:_1>

```

```

xmlns:action="http://nvd.nist.gov/ontology/ACTION/0.92#"
xmlns:organization="http://www.w3.org/2001/vcard-rdf/3.0#"
xmlns:for="http://nvd.nist.gov/ontology/FOR/0.92#"
xmlns:score="http://nvd.nist.gov/ontology/SCORE/0.92#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:product="http://nvd.nist.gov/ontology/PRODUCT/0.92#"
xmlns:topic="http://nvd.nist.gov/ontology/TOPIC/0.92#"
xmlns:vuln="http://nvd.nist.gov/ontology/VULNERABILITY/0.92#"
xmlns:configuration="http://nvd.nist.gov/ontology/CONFIGURATION/0.92#" >
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/PRODUCT/2">
  <product:TOPIC rdf:resource="http://nvd.nist.gov/ontology/data/TOPIC/2"/>
  <product:OWNER rdf:resource="http://www.redhat.com/">
  <product:TITLE>Red Hat Kernel</product:TITLE>
  <product:NAME>cpe:/a:redhat:kernel</product:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/TOPIC/1">
  <topic:PRODUCTS rdf:resource="http://nvd.nist.gov/ontology/data/PRODUCT/1"/>
  <topic:NAME>Application Server</topic:NAME>

```

```

<rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"/>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/TOPIC/2">
  <topic:PRODUCTS rdf:resource="http://nvd.nist.gov/ontology/data/PRODUCT/2"/>
  <topic:NAME>Kernel</topic:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/TOPIC/3">
  <topic:NAME>OS</topic:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/PRODUCT/1">
  <product:TOPIC rdf:resource="http://nvd.nist.gov/ontology/data/TOPIC/1"/>
  <product:OWNER rdf:resource="http://www.redhat.com/">
  <product:TITLE>Red Hat Application Server</product:TITLE>
  <product:NAME>cpe:/a:redhat:application_server</product:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://www.redhat.com/">
  <organization:NAME>Red Hat</organization:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/PRODUCT/3">
  <product:CONTAINS rdf:nodeID="A1"/>
  <product:TOPIC rdf:resource="http://nvd.nist.gov/ontology/data/TOPIC/3"/>
  <product:TITLE rdf:nodeID="A0"/>
  <product:VERSION>enterprise_linux</product:VERSION>

```

```
<product:OWNER rdf:resource="http://www.redhat.com"/>
<product:NAME>cpe:/o:redhat:enterprise_linux</product:NAME>
</rdf:Description>
<rdf:Description rdf:nodeID="A1">
<rdf:_1 rdf:resource="http://nvd.nist.gov/ontology/data/PRODUCT/2"/>
<rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"/>
</rdf:Description>
</rdf:RDF>
```

Step 9 - Another iteration is necessary to complete our requirements

Second Iteration

Step 1 - Describe your initial, most difficult requirements in conversational, informal English.

Leverage any existing diagrams or formalisms

We must track instances of products. An instance of a product is based on the definition of a product in conjunction with specific settings. Settings have domains and ranges for a set of values. These settings further describe products.

Policy refers to the process of making important organizational decisions, including the identification of different alternatives such as programs or spending priorities, and choosing among them on the basis of the impact they will have. Policies can be understood as political, management, financial, and administrative mechanisms arranged to reach explicit goals.

A security risk is classified as a vulnerability if it is recognized as a possible means of attack. A security risk with one or more known instances of working and fully-implemented attacks is classified as an exploit.

Vulnerabilities exist for products as a whole or with products instantiated with certain settings.

In computer systems a configuration is an arrangement of functional units according to their nature, number, and chief characteristics. Often, configuration pertains to the choice of hardware, software, firmware, and documentation. The configuration affects system function and performance.

Step 2 - Decompose the problem into domain components. Pick the most difficult domain as a starting point

The problem is now manageable. This is our final iteration. We will track Vulnerabilities, Configurations and Product Instances. At the conclusion of this domain analysis, the analysis is complete.

Step 3 - Look for opportunities of abstraction to lessen the number of components

Figure 5

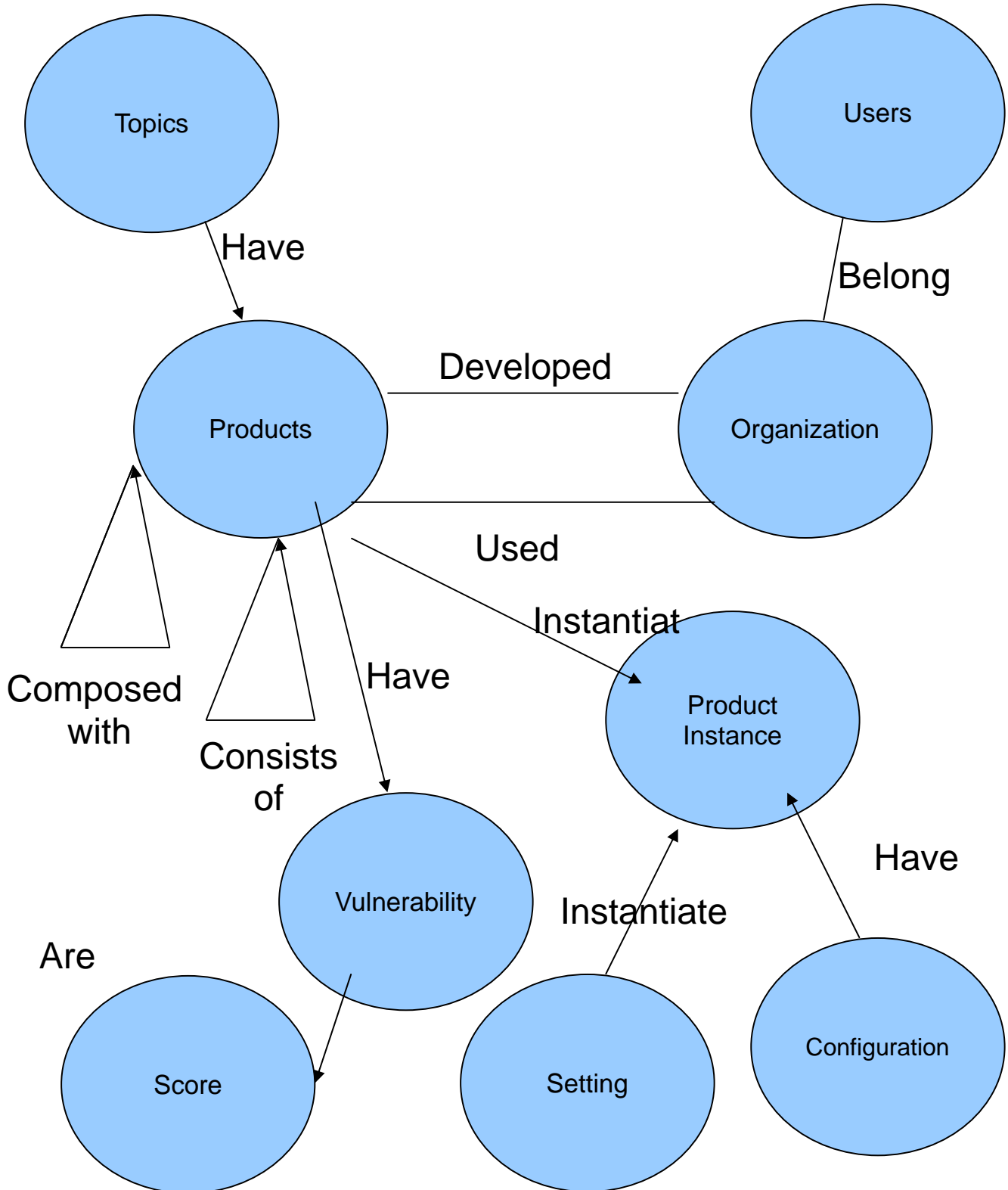
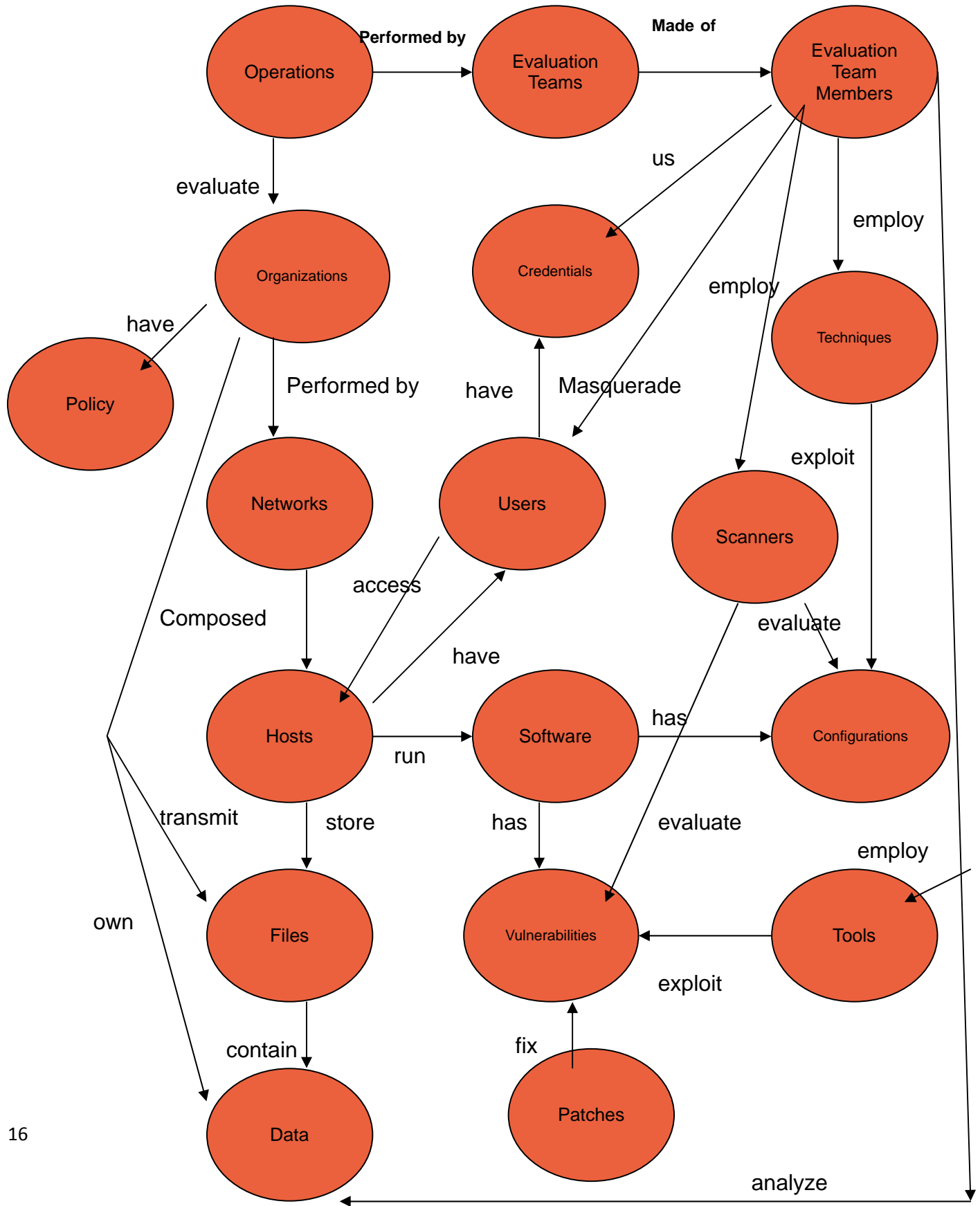


Figure 6



Step 4 - Research existing vocabularies and ontologies in similar domains to use in composition

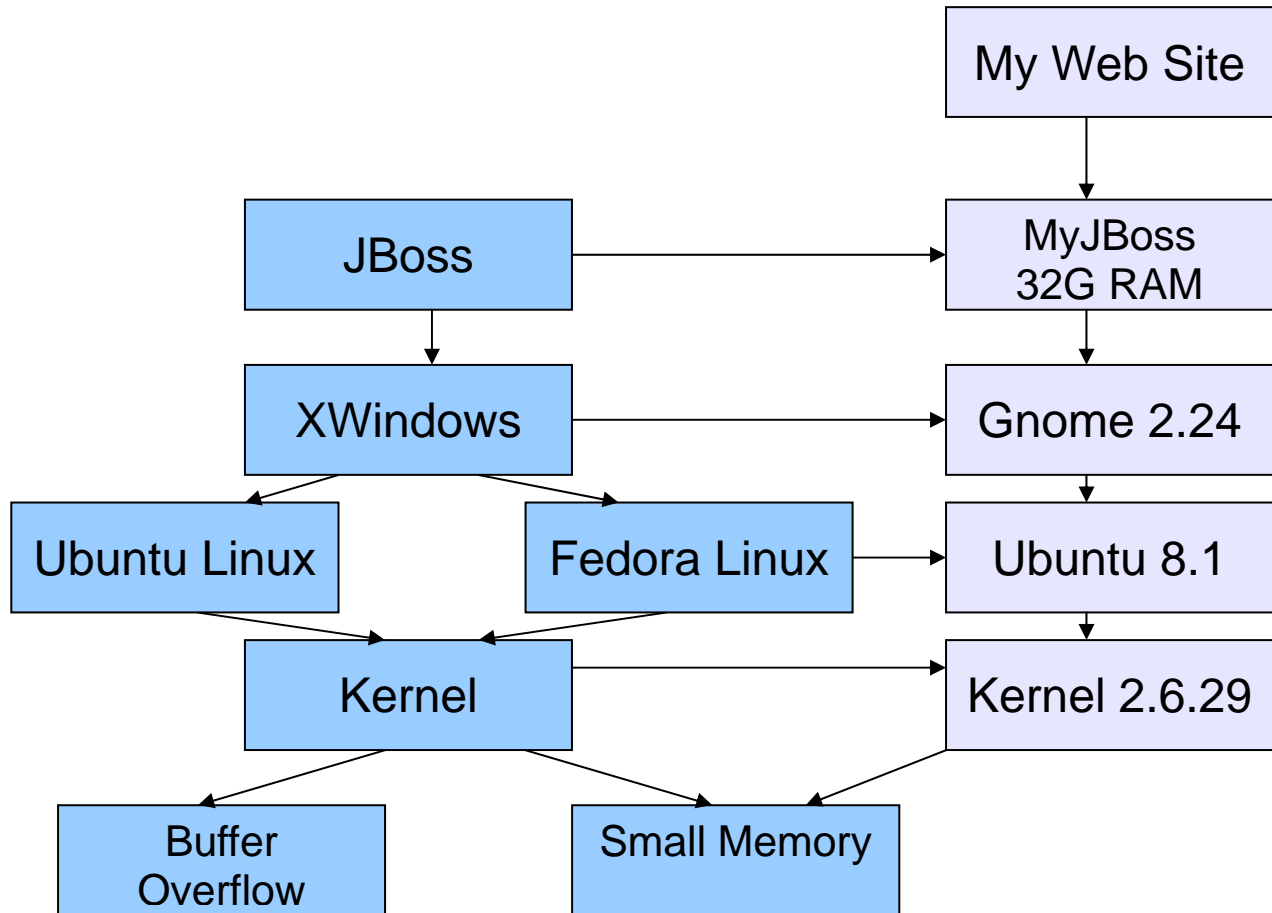
In the case of NIST, we use Dublin Core to represent Policy. It is a predefined vocabulary and a standard for representing publications. We may list or track policy using Dublin Core.

Step 5 - If a preexisting vocabulary does not exist, model it yourself creating Tbox entries

We create Tbox entries for Vulnerability, Configuration, Setting, Score, and Product Instance.

Step 6 - Take an instantiation of the data and prove it can work on paper

Figure 7



My web site runs on JBoss. JBoss runs on a Linux configuration. The Linux configuration, when running with less than 64 Gig of memory has a vulnerability of buffer overflow allowing compromised access to the system. Clearly our Tbox represents these instances. The definitions of products through composition are on the left in blue. The instances of product instance are on the right in grey.

Step 7 - Use a semantic web implementation, like Jena, to build a Tbox vocabulary

```
package semantic.exp;

import com.hp.hpl.jena.rdf.model.Model;

import com.hp.hpl.jena.rdf.model.ModelFactory;

import com.hp.hpl.jena.rdf.model.Property;

/**
 * This class represents a scored vulnerability associated
 * with a configuration (or just a bag) of products.
 */

public class VULNERABILITY {

    private static final String uri = "http://nvd.nist.gov/ontology/VULNERABILITY/0.92#";

    public static String getURI()

    {

        return uri;

    }

    private static Model m = ModelFactory.createDefaultModel();

    /** the ordered configuration of products associated with this vuln */

    public static final Property CONFIGURATION = m.createProperty(uri, "CONFIGURATION");

    /** the bag of products associated with vuln, not ordered in any way */

    public static final Property PRODUCT = m.createProperty(uri, "PRODUCT");

    /** this can be a patch (PRODUCT) or ACTION */

    public static final Property FIX = m.createProperty(uri, "FIX");
```

```
import com.hp.hpl.jena.rdf.model.Property;

/**
 * This represents a particular PRODUCT that is "instantiated" with a group of
 * all possible settings for the product, as well as a group of the actual
 * values of those settings within the context of this instance.
 */

public class PRODUCT_INSTANCE {

    private static final String uri = "http://nvd.nist.gov/ontology/PRODUCT_INSTANCE/0.92#";

    public static String getURI() {

        return uri;

    }

    private static Model m = ModelFactory.createDefaultModel();

    /** the instance of the product */

    public static final Property PRODUCT = m.createProperty(uri, "PRODUCT");

    /** the different possible settings */

    public static final Property SETTINGS = m.createProperty(uri, "SETTINGS");

    // TODO: need to have a way of mapping the actual values of the setting
    // to the setting in the bag

}
```

```

    /** this is a bag of all score attributes, can include anything */
    public static final Property SCORES = m.createProperty(uri, "SCORES");
}

package semantic.exp;

import com.hp.hpl.jena.rdf.model.Model;

import com.hp.hpl.jena.rdf.model.ModelFactory;

```

Step 8 - In a semantic web implementation, instantiate the vocabulary by creating instances and output the instances as RDF/XML

```

package semantic.exp;

import com.hp.hpl.jena.rdf.model.Bag;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.DC;
import com.hp.hpl.jena.vocabulary.VCARD;

public class CPEModelGenerator {
    public static void main(String args[]) throws Exception {
        Model model = ModelFactory.createDefaultModel();
        model.setNsPrefix("product", PRODUCT.getURI());
        model.setNsPrefix("topic", TOPIC.getURI());
    }
}

```

```

package semantic.exp;

import com.hp.hpl.jena.rdf.model.Bag;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.DC;
import com.hp.hpl.jena.vocabulary.VCARD;

public class CPEModelGenerator {
    public static void main(String args[]) throws Exception {
        Model model = ModelFactory.createDefaultModel();
        model.setNsPrefix("product", PRODUCT.getURI());
        model.setNsPrefix("topic", TOPIC.getURI());
    }
}

```

```

model.setNsPrefix("organization", VCARD.getURI());
model.setNsPrefix("action", ACTION.getURI());
model.setNsPrefix("vuln", VULNERABILITY.getURI());
model.setNsPrefix("score", SCORE.getURI());
model.setNsPrefix("dc", DC.getURI());
model.setNsPrefix("setting", SETTING.getURI());
model.setNsPrefix("for", FOR.getURI());
model.setNsPrefix("configuration", CONFIGURATION.getURI());

Resource redHat = model.createResource("http://www.redhat.com/");
redHat.addProperty(VCARD.NAME, "Red Hat");

Resource appServer = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/1");
appServer.addProperty(TOPIC.NAME, "Application Server");

Resource kernel = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/2");
kernel.addProperty(TOPIC.NAME, "Kernel");

Resource os = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/3");
os.addProperty(TOPIC.NAME, "OS");

Resource rhelAppServer =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/1");
rhelAppServer.addProperty(PRODUCT.NAME, "cpe:/a:redhat:application_server");

```

```

model.setNsPrefix("organization", VCARD.getURI());
model.setNsPrefix("action", ACTION.getURI());
model.setNsPrefix("vuln", VULNERABILITY.getURI());
model.setNsPrefix("score", SCORE.getURI());
model.setNsPrefix("dc", DC.getURI());
model.setNsPrefix("setting", SETTING.getURI());
model.setNsPrefix("for", FOR.getURI());
model.setNsPrefix("configuration", CONFIGURATION.getURI());

Resource redHat = model.createResource("http://www.redhat.com/");
redHat.addProperty(VCARD.NAME, "Red Hat");

Resource appServer = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/1");
appServer.addProperty(TOPIC.NAME, "Application Server");

Resource kernel = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/2");
kernel.addProperty(TOPIC.NAME, "Kernel");

Resource os = model.createResource("http://nvd.nist.gov/ontology/data/TOPIC/3");
os.addProperty(TOPIC.NAME, "OS");

Resource rhelAppServer =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/1");
rhelAppServer.addProperty(PRODUCT.NAME, "cpe:/a:redhat:application_server");

```

```

rhelAppServer.addProperty(PRODUCT.TITLES, "Red Hat Application Server");

rhelAppServer.addProperty(PRODUCT.OWNER, redHat);

rhelAppServer.addProperty(PRODUCT.TOPIC, appServer);

Resource rhelKernel = model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/2");

rhelKernel.addProperty(PRODUCT.NAME, "cpe:/a:redhat:kernel");

rhelKernel.addProperty(PRODUCT.TITLES, "Red Hat Kernel");

rhelKernel.addProperty(PRODUCT.OWNER, redHat); // is-a vendor

rhelKernel.addProperty(PRODUCT.TOPIC, kernel);

Resource rhel = model
.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/3");

rhel.addProperty(PRODUCT.NAME, "cpe:/o:redhat:enterprise_linux");

rhel.addProperty(PRODUCT.OWNER, redHat);

rhel.addProperty(PRODUCT.VERSION, "enterprise_linux");

Bag titles = model.createBag();

titles.add("Red Hat Enterprise Linux");

rhel.addProperty(PRODUCT.TITLES, titles);

rhel.addProperty(PRODUCT.TOPIC, os);

Bag container = model.createBag();

container.add(rhelKernel);

rhel.addProperty(PRODUCT.CONTAINS, container);

appServer.addProperty(TOPIC.PRODUCTS, rhelAppServer);

```

```

rhelAppServer.addProperty(PRODUCT.TITLES, "Red Hat Application Server");

rhelAppServer.addProperty(PRODUCT.OWNER, redHat);

rhelAppServer.addProperty(PRODUCT.TOPIC, appServer);

Resource rhelKernel = model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/2");

rhelKernel.addProperty(PRODUCT.NAME, "cpe:/a:redhat:kernel");

rhelKernel.addProperty(PRODUCT.TITLES, "Red Hat Kernel");

rhelKernel.addProperty(PRODUCT.OWNER, redHat); // is-a vendor

rhelKernel.addProperty(PRODUCT.TOPIC, kernel);

Resource rhel =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/3");

rhel.addProperty(PRODUCT.NAME, "cpe:/o:redhat:enterprise_linux");

rhel.addProperty(PRODUCT.OWNER, redHat);

rhel.addProperty(PRODUCT.VERSION, "enterprise_linux");

Bag titles = model.createBag();

titles.add("Red Hat Enterprise Linux");

rhel.addProperty(PRODUCT.TITLES, titles);

rhel.addProperty(PRODUCT.TOPIC, os);

Bag container = model.createBag();

container.add(rhelKernel);

rhel.addProperty(PRODUCT.CONTAINS, container);

appServer.addProperty(TOPIC.PRODUCTS, rhelAppServer);

```

```

kernel.addProperty(TOPIC.PRODUCTS, rhelKernel);
Resource google = model.createResource("http://www.google.com/");
    google.addProperty(VCARD.NAME, "Google");
Resource chrome = model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/88789");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:0.2.149.29");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles = model.createBag();
    chromeTitles.add("Google Chrome 0.2.149.29");
    chrome.addProperty(PRODUCT.TITLES, chromeTitles);
    chrome.addProperty(PRODUCT.VERSION, "0.2.149.29");
Resource chrome2
=model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/88790");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:0.2.149.30");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles2 = model.createBag();
    chromeTitles.add("Google Chrome 0.2.149.30");
    chrome.addProperty(PRODUCT.TITLES, chromeTitles2);
    chrome.addProperty(PRODUCT.VERSION, "0.2.149.30");
Resource chrome3 =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/94444");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:1.0.154.36");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles3 = model.createBag();
    chromeTitles.add("Google Chrome 1.0.154.36");

```

```

kernel.addProperty(TOPIC.PRODUCTS, rhelKernel);
Resource google = model.createResource("http://www.google.com/");
    google.addProperty(VCARD.NAME, "Google");
Resource chrome = model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/88789");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:0.2.149.29");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles = model.createBag();
    chromeTitles.add("Google Chrome 0.2.149.29");
    chrome.addProperty(PRODUCT.TITLES, chromeTitles);
    chrome.addProperty(PRODUCT.VERSION, "0.2.149.29");
Resource chrome2 =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/88790");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:0.2.149.30");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles2 = model.createBag();
    chromeTitles.add("Google Chrome 0.2.149.30");
    chrome.addProperty(PRODUCT.TITLES, chromeTitles2);
    chrome.addProperty(PRODUCT.VERSION, "0.2.149.30");
Resource chrome3 =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT/94444");
    chrome.addProperty(PRODUCT.NAME, "cpe:/a:google:chrome:1.0.154.36");
    chrome.addProperty(PRODUCT.OWNER, google);
Bag chromeTitles3 = model.createBag();
    chromeTitles.add("Google Chrome 1.0.154.36");

```

```

chrome.addProperty(PRODUCT. TITLES, chromeTitles3);

chrome.addProperty(PRODUCT. VERSION, "1.0.154.36");

Resource config1 =
model.createResource("http://nvd.nist.gov/ontology/data/CONFIGURATION/10");

Bag oredProducts = model.createBag();

oredProducts.add(chrome2);

oredProducts.add(chrome3);

config1.addProperty(CONFIGURATION. OR, oredProducts);

Resource vuln = model.createResource("http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-
2009-0411");

vuln.addProperty(VULNERABILITY. CONFIGURATION, config1);

Bag vulnScore = model.createBag();

Resource baseScore = model.createResource("http://nvd.nist.gov/ontology/data/SCORE/5base");

baseScore.addProperty(SCORE. SCORE, "5");

baseScore.addProperty(SCORE. SCORE_VECTOR, "base");

Resource accessVector =
model.createResource("http://nvd.nist.gov/ontology/data/SCORE/NetworkAccessVector");

accessVector.addProperty(SCORE. SCORE, "Network");

accessVector.addProperty(SCORE. SCORE_VECTOR, "AccessVector");

vulnScore.add(baseScore);

vulnScore.add(accessVector);

vuln.addProperty(VULNERABILITY. SCORES, vulnScore);

Resource setting = model.createResource("http://nvd.nist.gov/ontology/data/SETTING/1");

setting.addProperty(SETTING. DOMAIN, "firewall-state");

```

```

chrome.addProperty(PRODUCT. TITLES, chromeTitles3);

chrome.addProperty(PRODUCT. VERSION, "1.0.154.36");

Resource config1 =
model.createResource("http://nvd.nist.gov/ontology/data/CONFIGURATION/10");

Bag oredProducts = model.createBag();

oredProducts.add(chrome2);

oredProducts.add(chrome3);

config1.addProperty(CONFIGURATION. OR, oredProducts);

Resource vuln = model.createResource("http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-
2009-0411");

vuln.addProperty(VULNERABILITY. CONFIGURATION, config1);

Bag vulnScore = model.createBag();

Resource baseScore = model.createResource("http://nvd.nist.gov/ontology/data/SCORE/5base");

baseScore.addProperty(SCORE. SCORE, "5");

baseScore.addProperty(SCORE. SCORE_VECTOR, "base");

Resource accessVector =
model.createResource("http://nvd.nist.gov/ontology/data/SCORE/NetworkAccessVector");

accessVector.addProperty(SCORE. SCORE, "Network");

accessVector.addProperty(SCORE. SCORE_VECTOR, "AccessVector");

vulnScore.add(baseScore);

vulnScore.add(accessVector);

vuln.addProperty(VULNERABILITY. SCORES, vulnScore);

Resource setting = model.createResource("http://nvd.nist.gov/ontology/data/SETTING/1");

setting.addProperty(SETTING. DOMAIN, "firewall-state");

```

```

Bag settingRange = model.createBag();

    settingRange.add("on");

    settingRange.add("off");

    setting.addProperty(SETTING.RANGE, settingRange);

Resource productInstance =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT_INSTANCE/1");

    productInstance.addProperty(PRODUCT_INSTANCE.PRODUCT, chrome3);

Bag possibleSettings = model.createBag();

    possibleSettings.add(setting);

    productInstance.addProperty(PRODUCT_INSTANCE.SETTINGS, possibleSettings);

//capture the value for the setting

Resource captureValue = model.createResource("http://nvd.nist.gov/ontology/data/FOR/1");

    captureValue.addProperty(FOR.REFERENCE_URI, setting);

    captureValue.addProperty(FOR.VALUE, "on");

//capture an action

Resource action1 = model.createResource("http://nvd.nist.gov/ontology/data/ACTION/1");

    Resource testingOrg = model.createResource("http://www.sometestOrg.com");

    testingOrg.addProperty(VCARD.NAME, "ACME Testing Organizatoin");

    action1.addProperty(ACTION.PERFORMED_BY, testingOrg);

Bag possibleVulns = model.createBag();

    possibleSettings.add(vuln);

```

```

Bag settingRange = model.createBag();

    settingRange.add("on");

    settingRange.add("off");

    setting.addProperty(SETTING.RANGE, settingRange);

Resource productInstance =
model.createResource("http://nvd.nist.gov/ontology/data/PRODUCT_INSTANCE/1");

    productInstance.addProperty(PRODUCT_INSTANCE.PRODUCT, chrome3);

Bag possibleSettings = model.createBag();

    possibleSettings.add(setting);

    productInstance.addProperty(PRODUCT_INSTANCE.SETTINGS, possibleSettings);

//capture the value for the setting

Resource captureValue = model.createResource("http://nvd.nist.gov/ontology/data/FOR/1");

    captureValue.addProperty(FOR.REFERENCE_URI, setting);

    captureValue.addProperty(FOR.VALUE, "on");

//capture an action

Resource action1 = model.createResource("http://nvd.nist.gov/ontology/data/ACTION/1");

    Resource testingOrg = model.createResource("http://www.sometestOrg.com");

    testingOrg.addProperty(VCARD.NAME, "ACME Testing Organizatoin");

    action1.addProperty(ACTION.PERFORMED_BY, testingOrg);

Bag possibleVulns = model.createBag();

    possibleSettings.add(vuln);

```

```
action1.addProperty(ACTION.PERFORMED_ON, possibleVulns);

// output...
model.write(System.out);
model.write(System.out, "N-TRIPLE");

    }
}
```

```
action1.addProperty(ACTION.PERFORMED_ON, possibleVulns);

// output...
model.write(System.out);
model.write(System.out, "N-TRIPLE");

    }
}
```