Semantic Web Methodology

Semantic Web Methodolgy

- Steps for a Semantic Web Methodology
  - 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. Add easily understood domains to the solution.
  - Step 3: Look for opportunities of abstraction to lessen the number of components.

Semantic Web Methodology

- Steps for a Semantic Web Methodology
  - 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.

Semantic Web Technology

- Steps for a Semantic Web Methodology
  - 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.

Semantic Web Methodology

- National Institute of Standards Technology (NIST) Case Study
  - NIST agreed to be the domain holder for PRODUCT,CONFIGURATION,VULNERABILITY, TOPIC, PRODUCT-INSTANCE
  - They will provide web server setup that provides documentation for the namespace and URI
  - NIST will also manage versions and releases of the domain vocabularies

Semantic Web Methodology

- Step 1: Describe your initial, most difficult requirements in conversational, informal English
  - For 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:
    - Hardware: Hardware is the physical platform supporting the IT system. The type and model of hardware can be relevant for some guidance and vulnerabilities.
    - 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.
    - 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.

Semantic Web Methodology

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

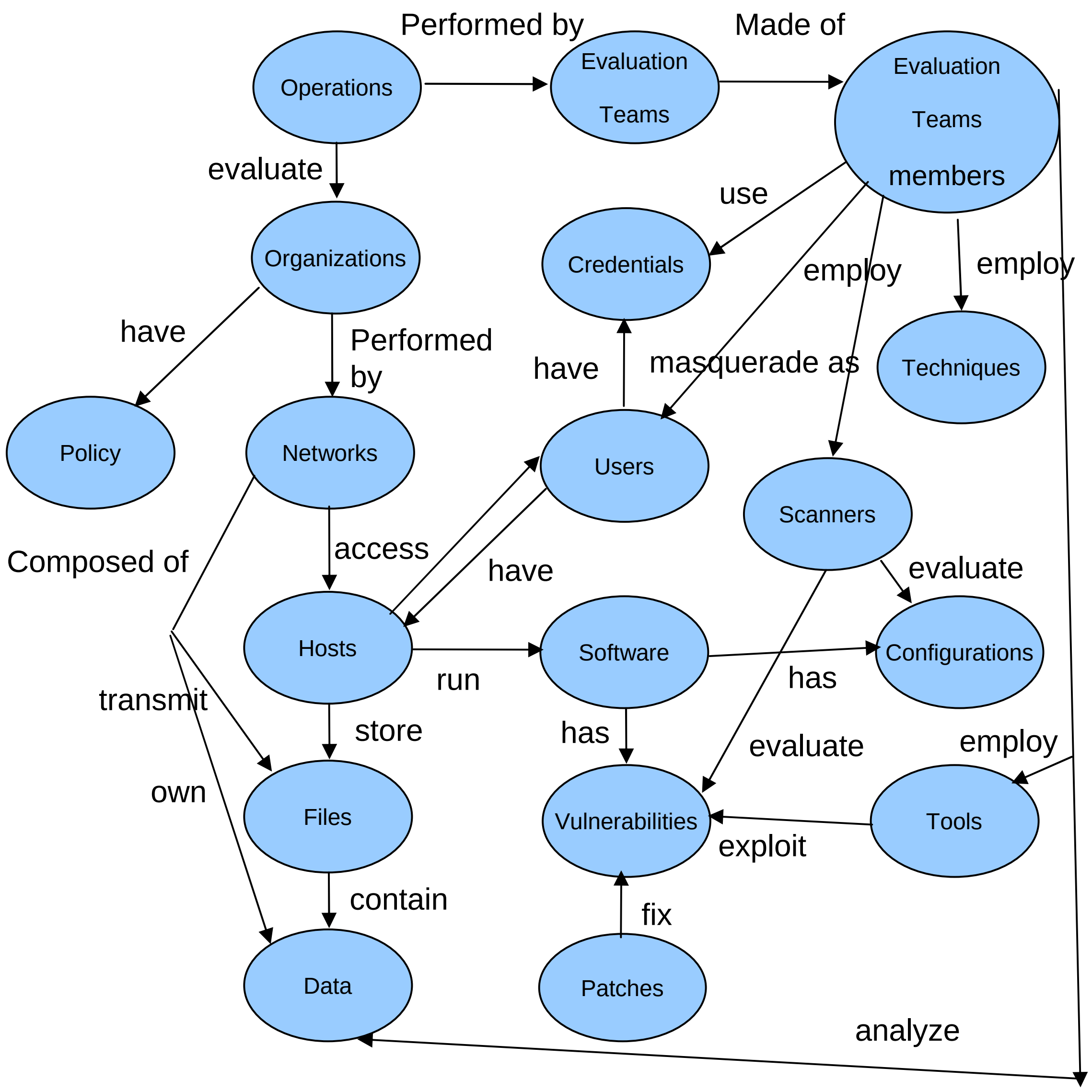
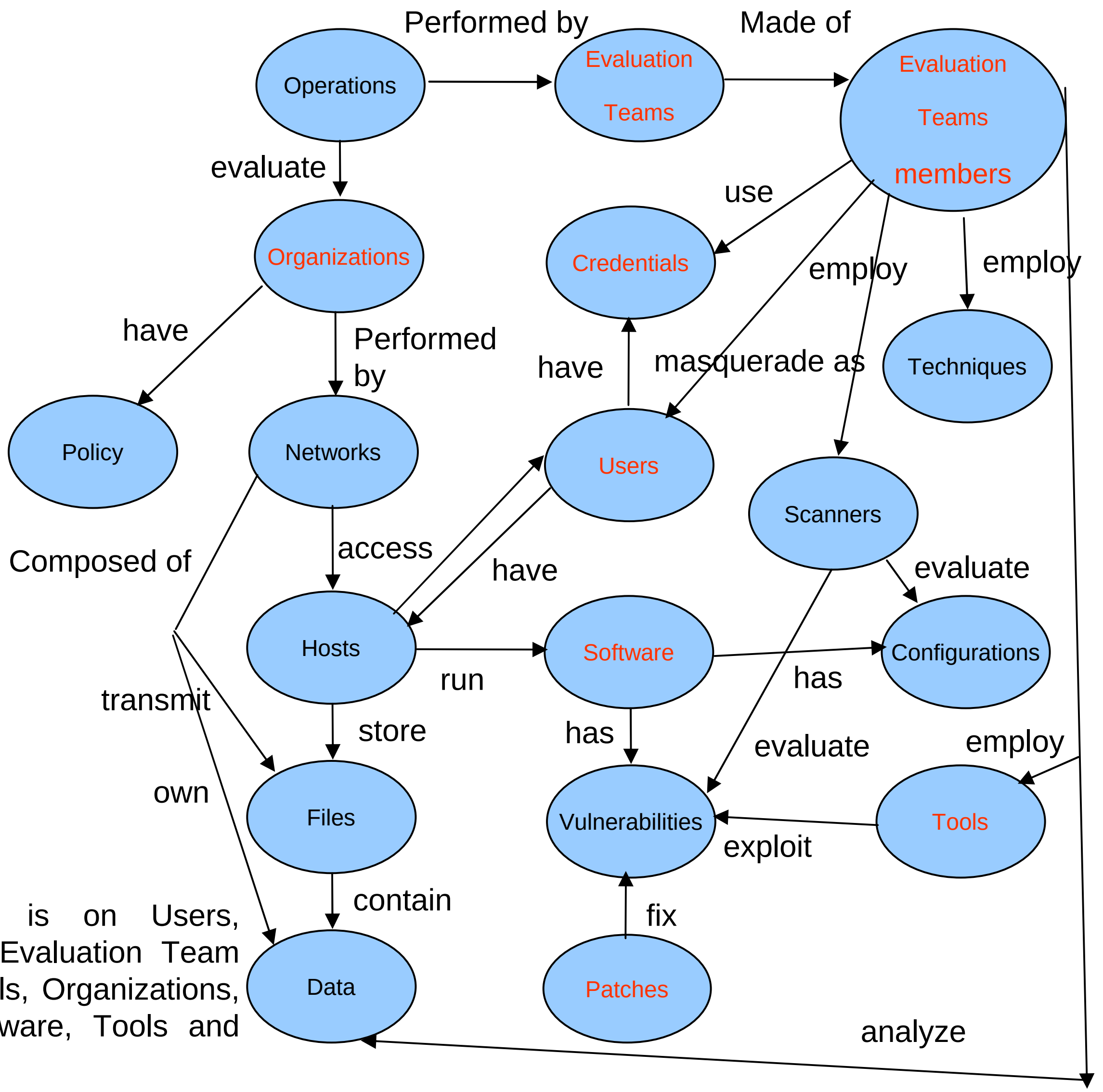


Figure 2 – Areas of Focus (Text in Red)



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

Semantic Web Methodology

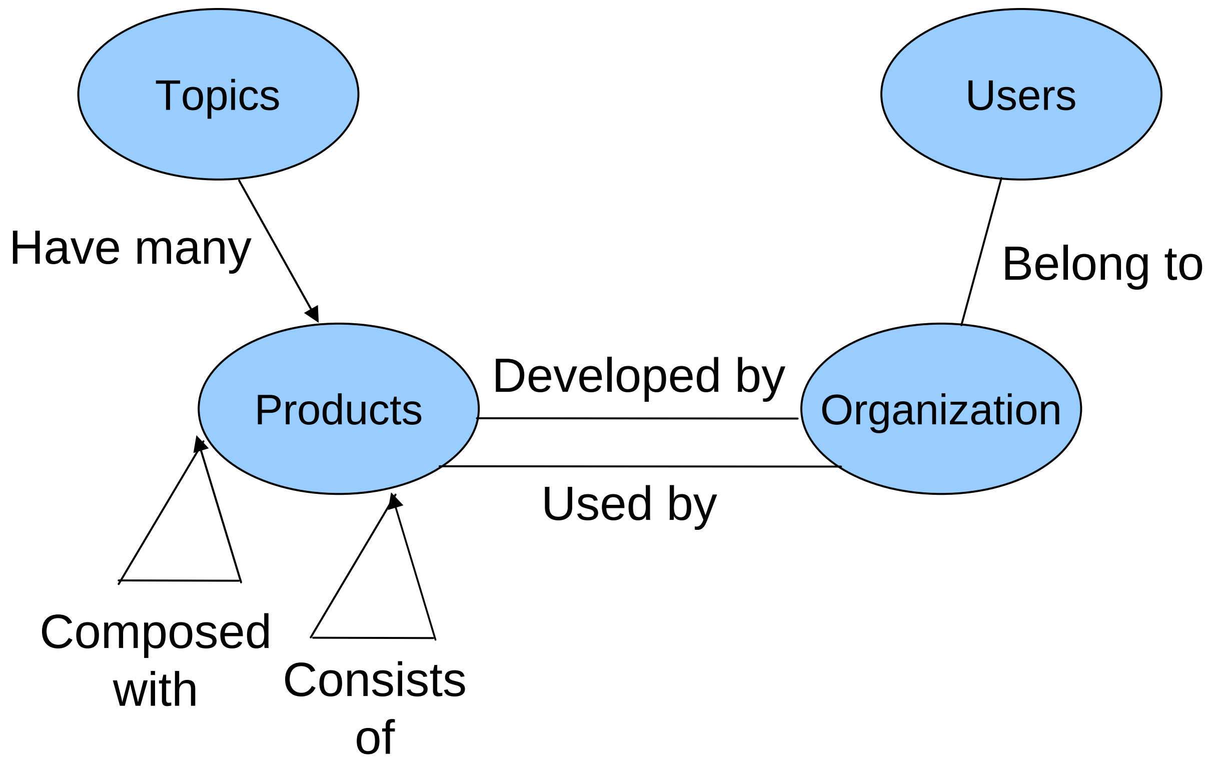
- Step 2: Decompose the problem into domain components. Pick the most difficult domain as a starting point
  - Clearly, the software/tools domain is the most complex. We add in Users, Evaluation Teams, Evaluation Team Members and Credentials since we know there are predefined Tbox for this domain and 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.

Semantic Web Methodology

- 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



Semantic Web Methodology

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

Semantic Web Methodology

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

Semantic Web Methodology

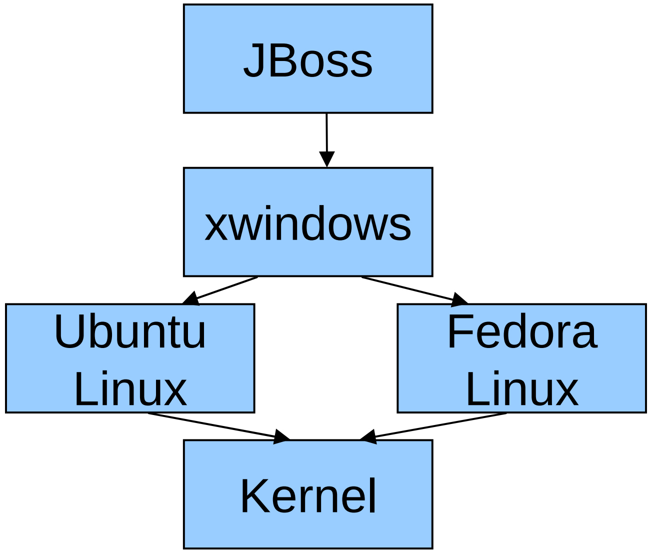
- Step 4 - Research existing vocabularies and ontologies in similar domains to use in composition
  - Friend Of A Friend (FOAF) and Virtual Card (VC) are considerations for Users and Organizations. We choose VC for better support of Organizations and the relationship to Users. No suitable domain vocabularies exist for the remainder of the problem so we need a custom vocabulary for Product and Topic.

Semantic Web Methodology

- 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.

Semantic Web Methodology

- Step 6: Take an instantiation 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.



Semantic Web Methodology

- Step 7: Use a Semantic Web implementation, like Jena, to build a Tbox vocabulary

```
public class NVD {
   /** <p>The ontology model that holds the vocabulary terms</p> */
   private static OntModel m_model = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM, null );

   /** <p>The namespace of the vocabulary as a string</p> */
   public static final String NS = "http://nvd.nist.gov/ontology.owl#";

   /** <p>The namespace of the vocabulary as a string</p>
    *  @see #NS */
   public static String getURI() {return NS;}

   /** <p>The namespace of the vocabulary as a resource</p> */
   public static final Resource NAMESPACE = m_model.createResource( NS );

   /** <p>Signifies this product is contained in the specified Product</p> */
   public static final ObjectProperty containedIn = m_model.createObjectProperty(
      "http://nvd.nist.gov/ontology.owl#containedIn" );

   /** <p>A product can contain more products.</p> */
   public static final ObjectProperty contains = m_model.createObjectProperty(
      "http://nvd.nist.gov/ontology.owl#contains" );
```

Semantic Web Methodology

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

```
public static void main(String args[]) throws Exception {
Model model = ModelFactory.createDefaultModel();
model.setNsPrefix("NVD", NVD.getURI());

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

Resource appServer = model.createResource(NVD.Topic.getURI() + "1");
appServer.addProperty(RDFS.label, "Application Server");

Resource kernel = model.createResource(NVD.Topic.getURI() + "2");
kernel.addProperty(RDFS.label, "Kernel");

Resource os = model.createResource(NVD.Topic.getURI() + "3");
os.addProperty(RDFS.label, "OS");
```

Semantic Web Methodology

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

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:NVD="http://nvd.nist.gov/ontology.owl#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
 <rdf:Description rdf:about="http://nvd.nist.gov/ontology.owl#Action1">
  <NVD:performedOn rdf:resource="http://nvd.nist.gov/ontology.owl#Vulnerability1"/>
  <NVD:performedBy rdf:resource="http://www.sometestOrg.com"/>
 </rdf:Description>
 <rdf:Description rdf:about="http://nvd.nist.gov/ontology.owl#Product3">
  <NVD:contains rdf:resource="http://nvd.nist.gov/ontology.owl#Product2"/>
```

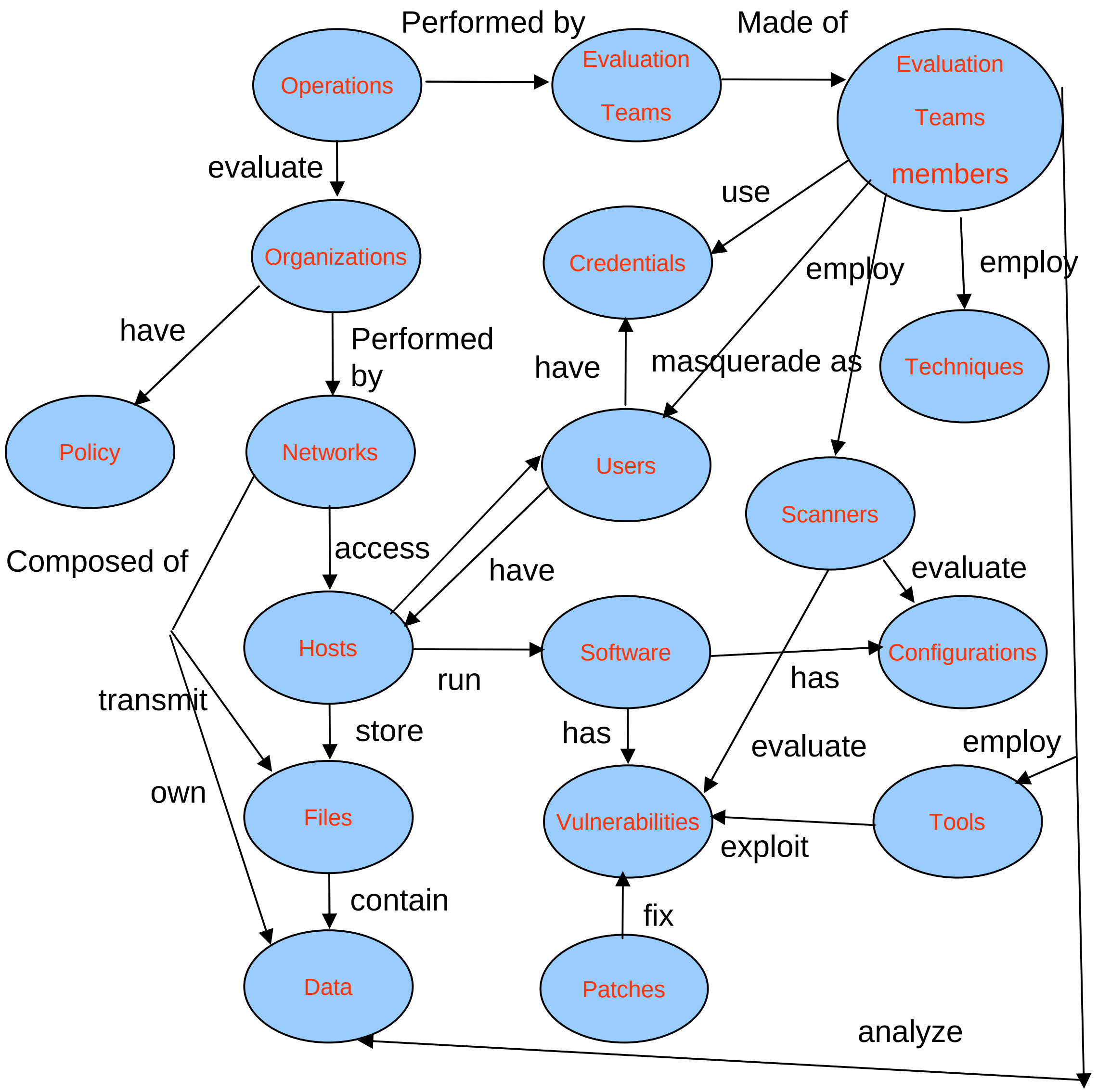
Semantic Web Methodology

- 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.

Semantic Web Methodology

- 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.

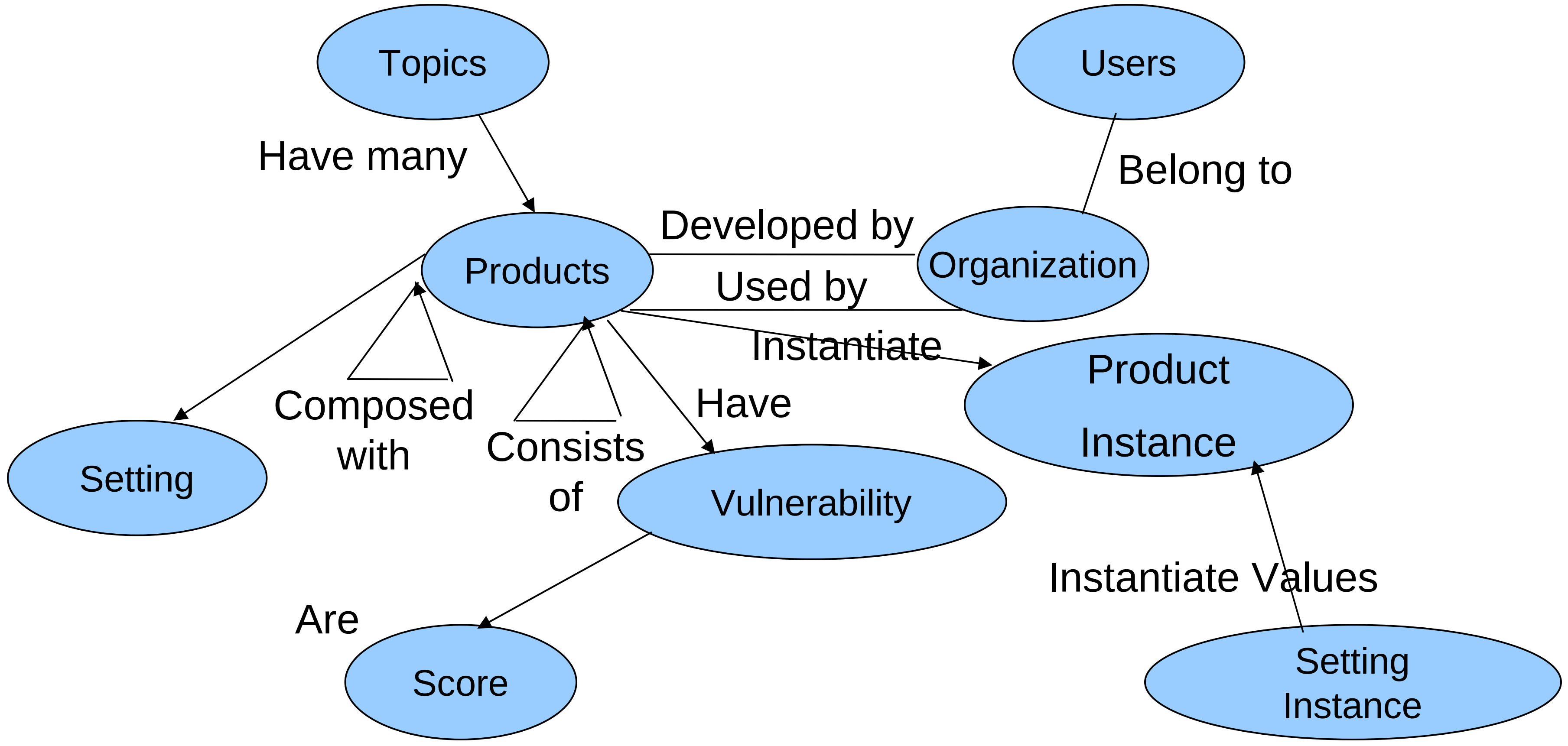
Figure – Areas of Focus (Text in Red)



Semantic Web Methodology

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

Figure 3 – An abstraction over the aforementioned red labeled notes



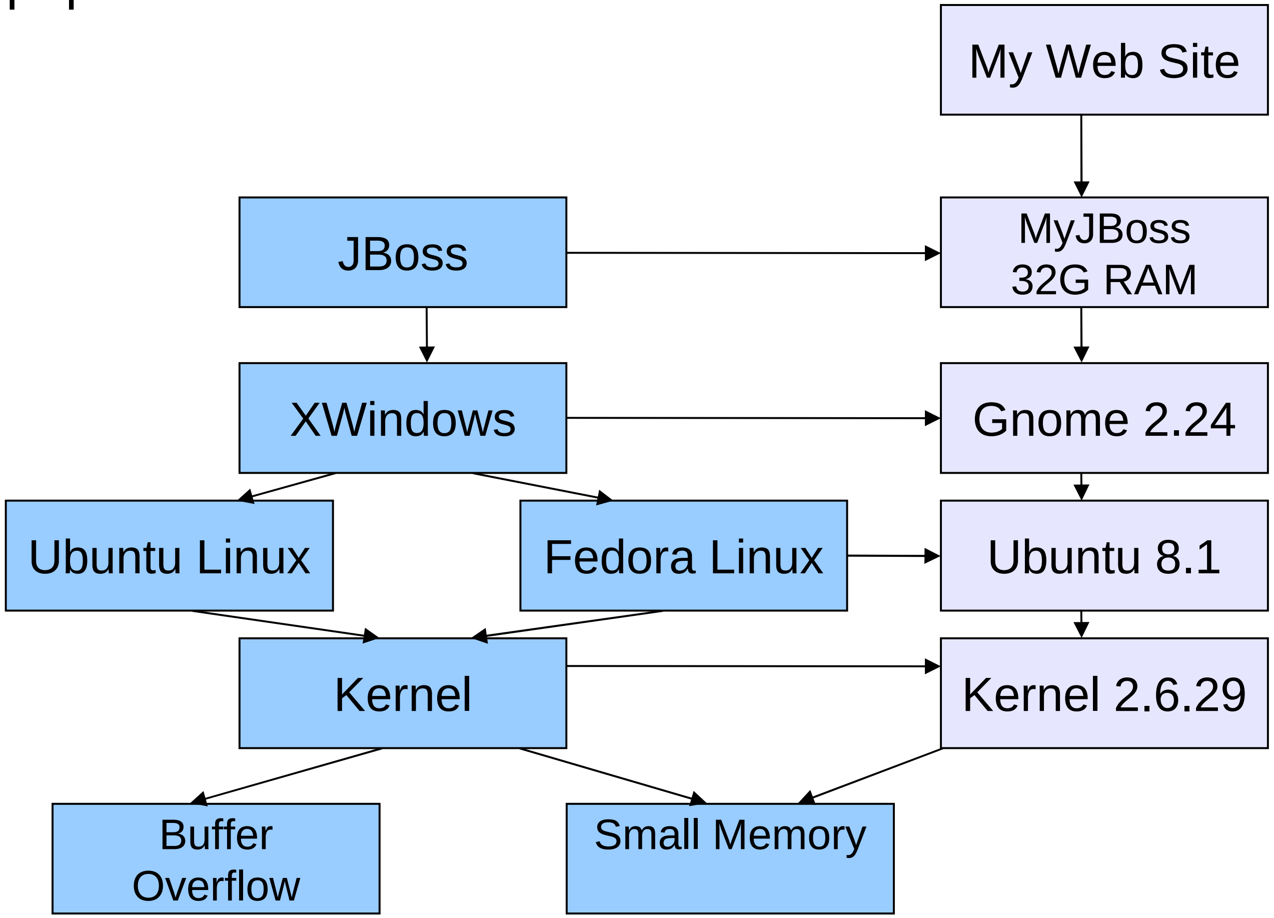
Semantic Web Methodology

- 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.

Semantic Web Methodology

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



Semantic Web Methodology

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

```
accessVector.addProperty(NVD.score, "Network");
accessVector.addProperty(NVD.scoreVector, "AccessVector");
vuln.addProperty(NVD.score, baseScore);
vuln.addProperty(NVD.score, accessVector);

// a setting present on a PRODUCT_INSTANCE or PRODUCT_FINDING
Resource setting = model.createResource(NVD.Setting + "1");
setting.addProperty(NVD.domain, "javascript_enabled");
setting.addProperty(NVD.range, "on");
setting.addProperty(NVD.range, "off");

// capture the value for the setting
Resource captureValue = model.createResource(NVD.SettingInstance
```

Semantic Web Methodology

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

```
<rdf:Description rdf:about="http://nvd.nist.gov/ontology/data/SCORE/NetworkAccessVector">
  <NVD:scoreVector>AccessVector</NVD:scoreVector>
  <NVD:score>Network</NVD:score>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology.owl#Product5">
  <NVD:version>6.92</NVD:version>
  <NVD:title xml:lang="EN">Microsoft IE 6</NVD:title>
  <NVD:owner rdf:resource="http://www.microsoft.com/"/>
  <NVD:alternateName>cpe:/a:microsoft:ie:6.92</NVD:alternateName>
</rdf:Description>
<rdf:Description rdf:about="http://www.sometestOrg.com">
  <vcard:NAME>ACME Testing Organizatoin</vcard:NAME>
</rdf:Description>
<rdf:Description rdf:about="http://nvd.nist.gov/ontology.owl#Setting1">
  <NVD:range>off</NVD:range>
  <NVD:range>on</NVD:range>
  <NVD:domain>javascript_enabled</NVD:domain>
```