NPS-AM-10-040

EXCERPT FROM THE

PROCEEDINGS

OF THE

SEVENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM WEDNESDAY SESSIONS VOLUME I

Acquisition Research

Creating Synergy for Informed Change

May 12 - 13, 2010

Published: 30 April 2010

Approved for public release, distribution unlimited.

Prepared for: Naval Postgraduate School, Monterey, California 93943

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Ontology-based Software Repository System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Systems Engineering Department,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
7th Annual Acquisition Research Symposium to be held May 12-13, 2010 in Monterey, California

14. ABSTRACT

The reuse of software and related artifacts is a key tenant of DoD acquisition improvement initiatives, including the Naval Open Architecture program. While there are many inhibitors of reuse, software repositories are considered enablers in that they provide a central store of artifacts as well as capabilities for search, retrieval, and reconfiguration of existing components into newly developed systems. However, current software repositories lack robust search and discovery capabilities and are thus limited enablers. This research expands on previous efforts to reform the organizational approach to software repositories by using ontologies as the framework of repository information. By combining metadata with domain, architectural, and other information, more sophisticated search techniques are enabled. In this paper, we describe the approach and demonstrate through a use case, a new type of search that takes advantage of the context provided by the ontologies and emphasizes human interaction. New navigation techniques will be employed that guide human users, offering suggestions based on projected needs. The improved search capability will encourage developers to consider reuse and improve the software reuse enabling power of software repositories.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**49**	

The research presented at the symposium was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret.)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.net

Ontology-based Software Repository System

Jean Johnson—Jean Johnson is a Lecturer in the Systems Engineering Department at the Naval Postgraduate School, Monterey, California. After serving on active duty in the US Navy, she supported the NAVSEA Warfare Systems Engineering Directorate (NAVSEA06) before coming to Naval Postgraduate School. Her current research focus areas are software repositories to enable reuse and the use of modeling and simulation in DoD acquisition. She is currently a PhD candidate in Software Engineering.

Jean M. Johnson
Systems Engineering Department
Naval Postgraduate School
Monterey, CA
e-mail: jmjohnso@nps.edu

Abstract

The reuse of software and related artifacts is a key tenant of DoD acquisition improvement initiatives, including the Naval Open Architecture program. While there are many inhibitors of reuse, software repositories are considered enablers in that they provide a central store of artifacts as well as capabilities for search, retrieval, and reconfiguration of existing components into newly developed systems. However, current software repositories lack robust search and discovery capabilities and are thus limited enablers.

This research expands on previous efforts to reform the organizational approach to software repositories by using ontologies as the framework of repository information. By combining metadata with domain, architectural, and other information, more sophisticated search techniques are enabled. In this paper, we describe the approach and demonstrate, through a use case, a new type of search that takes advantage of the context provided by the ontologies and emphasizes human interaction. New navigation techniques will be employed that guide human users, offering suggestions based on projected needs. The improved search capability will encourage developers to consider reuse and improve the software reuse enabling power of software repositories.

Introduction

Architectural and other information about software systems may be captured in a domain-specific ontological framework for a software repository to enable new types of searches. The relations in the ontology are used to determine associations between repository artifacts to facilitate intuitive navigation. A fisheye graph view enables visualization of artifacts within a contextual framework that provides suggestions based on users' actions. The emphasis is to provide a rich human interface that maximizes the combined knowledge of both the community of human users and the computer-based repository system. Capturing ontologies in standard formats results in an extensible framework, which can easily be shared between multiple repositories using XML-based technologies, thereby improving interoperability.

The initial target of the research is the US Navy's Software, Hardware Asset Reuse Enterprise (SHARE) repository. In 2007, researchers at the Naval Postgraduate School (NPS) were tasked to develop a component specification and ontology for the SHARE

software repository, which was then recently established. The component specification and ontology provide a rich structural and semantic framework for SHARE that enables multiple kinds of search and discovery techniques. Follow on work was assigned to develop designs for a software repository tool that will fully utilize the repository framework to improve the usefulness of SHARE. This paper captures the results of the effort with the intention of illustrating how the approach can be extended to additional applications and domains.

Current State of the Art

Improvements to the current state of the art for software reuse repositories are required (Shiva & Shala, 2007). Current software reuse repositories such as SourceForge (http://sourceforge.net) and Comprehensive Perl Archive Network (CPAN) (http://www.cpan.org) typically enable search and discovery of software artifacts through keyword searches over metadata or browsing through metadata via broad categories of artifacts (i.e., faceted classification) (Guo & Luqi, 2000). This approach is only effective in relatively small repositories or in situations where the users are well familiar with the contents of the repository. This is because successful discovery depends on the searchers' ability to express the desired results in the same vocabulary used by the artifact submitter or repository manager. In other words, if you know exactly what you are looking for, and how to ask for it, you will find it.

Search tools are not designed to aid users that do not already know the desired outcome of the search. A repository interface should guide searchers through the discovery process based on the users' context, suggesting search threads and recommending items for retrieval. However, current repositories do not support this type of repository navigation.

Current repositories in general do not relate artifacts to any context other than characterizations identified within the metadata. These typical characterizations enable a list-type search result, similar to the popular children's card game "Go Fish." Users can request, "Give me all of your artifacts of type xyz." Unfortunately, if you don't know what you want, you cannot ask for it. If you ask for artifacts of type xyz and there are no artifacts labeled as such in the repository, the search ends (go fish). Without contextual linkages between artifacts in the repository, guided searches are not possible. But, with a guided search, the system can recommend other potential solutions based on a given context.

Proposed Improvements

This research aims to produce a new kind of software repository that addresses the current shortfalls. There are four design characteristics that constitute the novelty of the approach.

First we take a broad view of reuse. Although reusable software artifacts are often defined to include any product related to the development of software, typical software repositories enable only the reuse of code or executable files and maybe some architecture and design products. We consider all types of artifacts from the software engineering life cycle—including requirements, test scripts, etc.—and plan for them in the design.

Second, we propose the use of ontologies to provide the contextual framework for the repository. We will show how these ontologies can be used to guide users to discover artifacts that they may find useful.

Third, we will exploit the use of domain-specific information in our repository design. By narrowing the reuse efforts to a particular domain, we increase our likelihood of

developing a repository that will be relevant to the user group. In the same way that successful strategic reuse has most often been associated with a domain-centric or product line development approach, a reuse repository is likely to be most successfully employed if it embodies, and is limited to, the domain knowledge base.

Finally, we propose the use of multiple views that will allow the user to view the repository contents in a comfortable visual arrangement for their particular use, depending on the experience of the user. The multiple views approach is analogous to Kruchten's multiple views of software architecture as depicted in his famous "4+1" paper (Kruchten, 1995).

Paper Organization

The remainder of the paper is organized as follows. In section two, we describe the proposed repository system. In section three, we provide a use case demonstration of the repository functionality. Sections four and five provide discussions of relevant related work, a summary, and suggested related future work.

A New Repository Tool

We propose the development of a new software repository tool that will encourage improved (increased and more effective) reuse of software artifacts by presenting information about the contents of the repository to the user in ways that allow the user to project their individual context onto the repository. Here we consider the user's context to include their instantaneous progress in the development process at the time of search, the system modeling paradigms with which they are comfortable (e.g., UML or DODAF), and their understanding of the particular domain for which they are developing systems.

In this section we will present the key features of the new repository tool, including an explanation of what is meant by the guided search and descriptions of the proposed ontology-based repository framework and envisioned visualization techniques.

Guided Search

The tool will support the human user by enabling smart navigation of the repository contents based on information collected. It is important to note that we can never completely automate the search process if we intend to incorporate unique user situations into the search algorithms. Therefore, the tool we suggest will guide the user through a search but cannot complete the search on its own. This is an important claim since the resulting necessary interaction between human and machine is an essential feature of the tool.

We envision a graphical "point and click" user interface that enables navigation of repository contents reflecting user interests. This requires an interface that allows users to project their context onto the search mechanisms. In other words, the users bring particular information needs and goals based on the problem they are trying to solve. For example, users may seek particular functionality best obtained through a functional organization of the information in the repository; they may seek particular artifacts best obtained through a document resource organization of the information; or, they may seek information on certain testing methodologies that have been applied so that a work activity organization of the information would best apply.

In our approach, relationships among assets and artifacts are recorded in the repository ontology framework, allowing users to navigate to and select artifacts based on their various relationships to an item of interest. Simple questions are posed of the user to initiate and direct the search at key moments. The items that are shown most prevalently to the user are determined by a prioritization scheme that takes into account previous user actions. Additionally, the user has the ability to "turn off" the relationships that are not helpful for the current search objectives. These capabilities are demonstrated in more detail for the SHARE repository in section three.

Repository Framework

In this section, we describe the framework for the repository that enables the functionality described. In (Johnson & Blais, 2008), we proposed two major aspects for this framework: a component specification and ontology. The component specification is a description or model of the items in the repository and consists of both typical metadata and a behavioral model of the component. The ontology describes concepts and relationships to create various perspectives or contexts for examining the contents of the repository. These aspects of the framework are discussed further below.

Component Specification—Metadata

The metadata for each artifact should incorporate all necessary data for discovery and implementation. The metadata will aid repository users in determining if the item is suited for their use and will provide information about how to use the asset when it is retrieved. We refer to the "standard" or "typical" metadata since there are many existing examples of metadata similar in concept to that developed for the SHARE repository. The intent of the metadata is to describe artifacts and assets contained in the repository in sufficient detail to aid the repository user in determining if the item is worth retrieving for a particular use.

To be clear, we must provide our definition of two terms. The Navy Open Architecture (OA) program has adopted similar definitions for asset and artifact as those used in the Object Management Group (OMG) Reusable Asset Specification (RAS). In the RAS, artifacts are defined as "any work products from the software development lifecycle," and assets are a grouping of artifacts that "provide a solution to a problem for a given context" (Object Management Group, 2005). Accordingly, the RAS describes an approach for packaging artifacts into an asset. This is consistent with the current SHARE approach and remains consistent in the proposed metadata XML schema described here. Artifacts are described individually and the asset description consists of the listing of artifacts included for that asset along with some descriptive information (see Figure 1).

The artifacts schema is designed to be flexible in its implementation. All elements, types, and attributes in the schema are defined globally to facilitate reuse. The root element, *Artifacts*, is simply a container for any number of artifacts contained in a single instance of the schema. A specific artifact can be incorporated into the file in one of three ways—by providing the full artifact description or by reference, either to a physical location or by URL.

The guts of the artifact metadata are captured in the *ArtifactDescription* sub-element of the full artifact description. The information necessary to describe the artifacts differs depending on whether the artifact is software code or some other type. Therefore, the schema allows a choice between two types of artifact descriptions as shown in Figure 2. The *NonCodeDescription* element applies to any artifact not considered software code. The group of elements contained therein (shown in Figure 3) is also required for artifacts that fall

under the *CodeDescription* element category, but additional elements are required for code artifacts. For brevity, we have presented a small subset of the schema developed. Detailed descriptions of each element of the SHARE metadata schema are available in (Johnson & Blais, 2008).

While much of the metadata described is lacking in novelty, a subset of the elements identified as part of the *NonCodeDescription* element begins to reveal the unique approach we have developed. First, the *ArtifactType*, *ApplicableSystems*, and *ObjectiveArchitectureTags* all serve the specific purpose of relating individual repository artifacts to the ontological framework described later in the section. Second, the *SoftwareBehaviorDescription* element is a specific focus of the design. Since this piece of the component specification is not commonly incorporated into repositories in a standardized manner, we feel it is a specific focus area to identify the appropriate representation mechanisms for software behavior in the repository context.

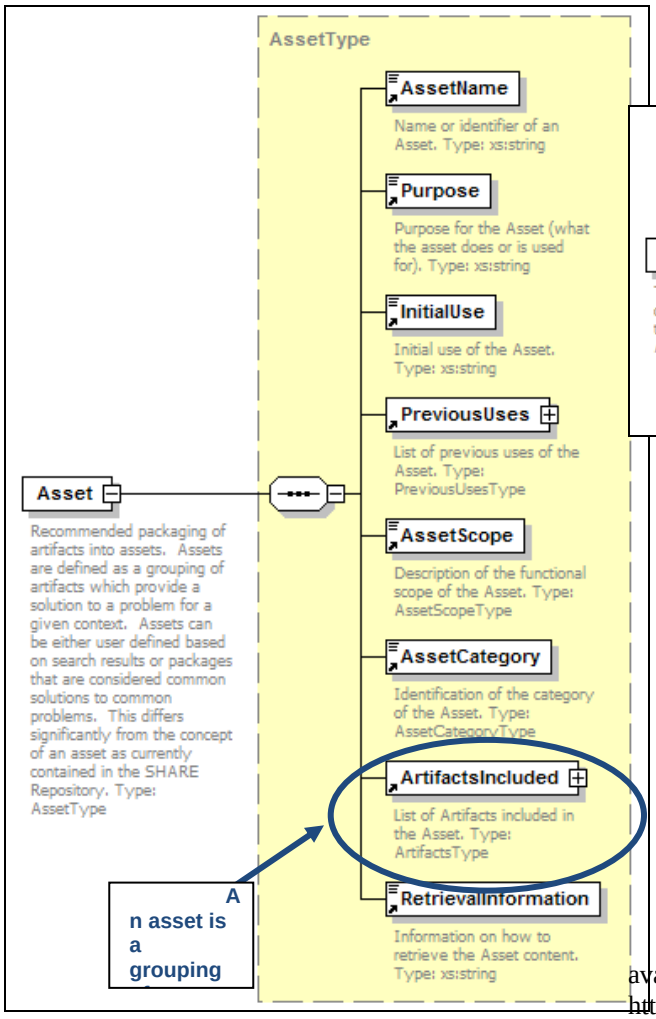


Figure 1. Asset Element[1]

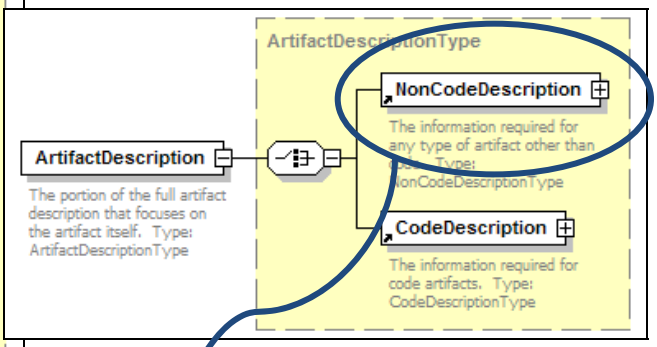


Figure 2. Artifact Description Element

[1] Diagrams of the XML structures have been generated by Altova XML-Spy. The product is available at http://www.altova.com/products/xmlspy/xml_editor.html. Altova offers a free 30-day license for trial use of the product. The Altova presentation of elements incorporates a plus (+) symbol on the right edge of a box to indicate that the element contains sub-elements.

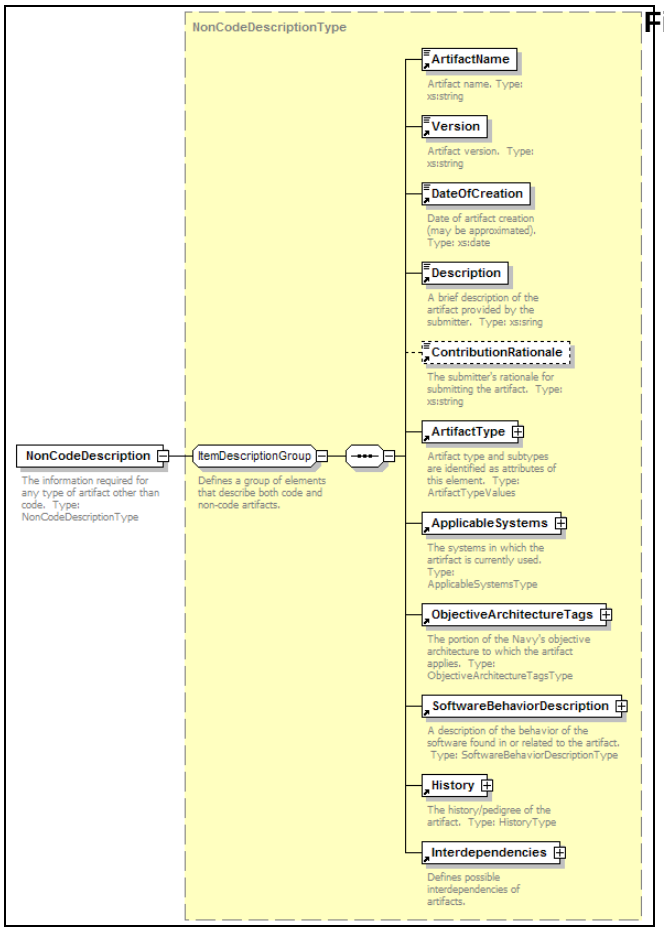


Figure 3. NonCodeDescription Element

Component Specification—Software Behavior Description

One of the loftier goals of a software repository is to support automatic composition of systems from reusable components. This is a difficult problem, which many have tried to solve.[2] It is especially difficult if the components were not originally designed for reuse. As a necessary first step towards more sophisticated uses of a repository, behavioral descriptions must be machine readable in order to support automated search and discovery. Furthermore, the behavior descriptions must be formalized and consistently applied to each item in the repository if the intent is to automatically compose them into a larger functioning system.

The array of contributors to SHARE and non-homogeneity of the repository contents requires caution in dictating standards that impact the development processes of the asset developers. It would not, for example, make sense to insist on a specific component technology across all Navy software programs in order to standardize the interface protocols. Yet, this is the type of precision required to truly enable software composition from reusable components. Recognizing that we fall short of this goal for this phase of the effort,

[2] The proceedings from the International Symposium on Software Composition, an annual event, provide examples of research into the breadth of research topics currently being pursued in the area of software composition. The web site for the 2008 conference is located at http://www.2008.software-composition.org/.

we have sought a balance between method robustness and ease of implementation in our software behavior specification.

In our approach, component behavior is potentially captured in two ways (as shown in Figure 4). First, the functionality of the software related to the artifact is identified by a list of functions selected from the Navy's Common System Function List (CSFL). The Navy's CSFL is a listing of functions that are performed by Navy systems. It provides a standardized taxonomy for the functionality found in this application domain. We have converted the CSFL into an ontology expressed in the Web Ontology Language (OWL), and acceptable entries for the *CommonSystemFunction* metadata element are validated against this ontology. If we require asset submitters to state the functionality of the components in these terms, we can then build the tools to guide users in selecting desired behavior in the same terms.

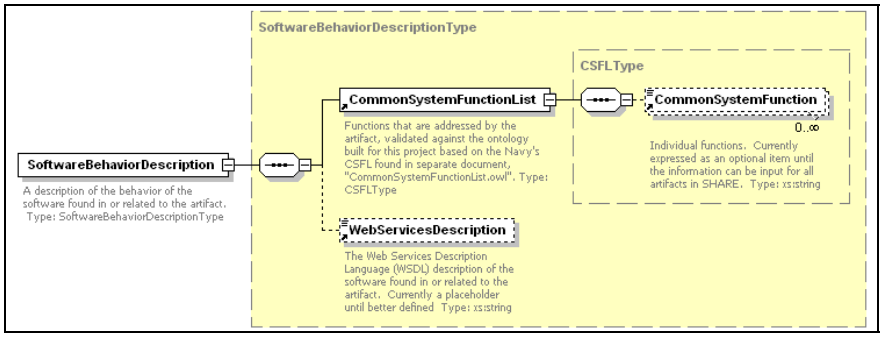


Figure 4. SoftwareBehaviorDescription Element

Second, the interface information may be captured as a Web Service Description Language (WSDL) document. In this research, we explored characterization of software interfaces based on current and emerging Web Services (e.g., WSDL) and Semantic Web Services (e.g., WS-BPEL, OWL-S) approaches. The work is preliminary, and it will be necessary to adopt a more precise description of code artifacts to introduce these techniques. As a start, we included the option of inserting a WSDL description of software services in the *SoftwareBehaviorDescription* element.

As the DoD moves toward Service Oriented Architectures (SOA), services may become a more frequent part of the SHARE repository. In that case, the WSDL describing those services (often automatically generated by the software development or execution environment of modern software systems) can be directly utilized in the repository to provide a detailed view of the service interfaces and operations. For software that is not developed and deployed as services, it is still feasible for public methods within the software to be parsed automatically to create WSDL-like descriptions. These may be incomplete descriptions with respect to full compliance to WSDL structures, but could still provide a well-defined way to describe the software for search and discovery.

Ontology Framework

The ontology framework provides contextual semantics that describe relationships among items in the repository to aid in associating artifacts with users' needs. It includes descriptions of the relationships of the components to form a contextual model of the repository items.

The taxonomies/ontologies we developed for SHARE are based on several types of relationships between the items in the repository, as well as with relevant domain

architectural descriptions and other information. They capture an artifact's place in the software engineering lifecycle, its architectural fit in its original system, its architectural fit in any system in which it was subsequently used, and identification of the component's fit in the Surface Navy Objective Architecture. Each of these ontologies is described in this section.

Software Lifecycle-Artifact Ontology

The software lifecycle-artifact ontology relates software artifacts to activities in the software engineering lifecycle. Aside from the "has subclass" relationship that exists in the software artifacts and lifecycle activities taxonomies, there are four additional properties that link these class structures:

- mayProduceArtifact—For each lifecycle activity, identifies which artifacts are most commonly produced as a result of that activity. The inverse property is oftenDevelopedDuring. The property maps items in the LifecyclePhases class (domain of the property) to the SoftwareArtifact class (range of the property).
- oftenDevelopedDuring—For each artifact, identifies the activity or activities that most commonly produce it. The inverse property is mayProduceArtifact. The property maps items in the SoftwareArtifact class (domain) to the LifecyclePhases class (range).
- mayRequireUseOf—For each lifecycle activity, identifies the most commonly needed artifacts. The inverse property is oftenUsedDuring. The property maps items in the LifecyclePhases class (domain) to the SoftwareArtifact class (range).
- oftenUsedDuring—For each artifact, identifies the activity or activities in which it is most commonly needed. The inverse property is mayRequireUseOf. The property maps items in the SoftwareArtifact class (domain) to the LifecyclePhases class (range).

To demonstrate, a diagram showing the relations captured for the RequirementsSpecification and RequirementsDatabase classes of the software artifact taxonomy are shown in Figure 5.

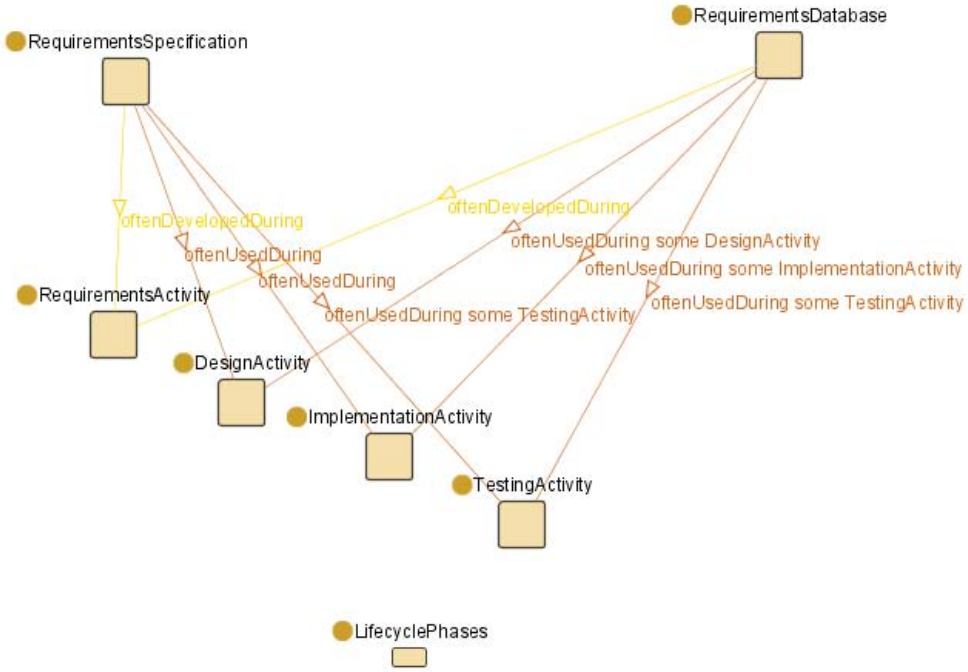


Figure 5. Properties Assigned to RequirementsSpecification and RequirementsDatabase Classes (developed using Jambalaya tab in Protégé)[3]

Objective Architecture Taxonomy

This taxonomy represents the decomposition of the common architecture for Navy combat systems, and was built directly from the already existing Surface Combat System Top-Level Objective Architecture. The taxonomy enables the repository system to correlate artifacts that have similar relationships based on commonality within the architecture to suggest them as possible items for retrieval.

System-SubSystem Ontologies

Here we provide one example (Figure 6) of how systems/subsystems and their interfaces can be captured as an ontology to complement the repository framework. Our recommendation is that ontologies be developed to capture each of the systems contained in the repository. As mentioned previously, the system/subsystem taxonomies would be used to verify the entries for the *System* and *Subsystem* elements in the metadata in order to assign artifacts to classes and subclasses (as individuals) within the ontology. Once these are assigned, the repository application could derive interface and other relationships from the ontology.

Each piece of the repository framework enhances the search capabilities in different ways. The basic metadata in the XML schemas provide search criteria for finding components of interest in the repository as well as specific information about the artifacts to determine if they are appropriate for retrieval. OWL taxonomies and ontologies enable

[3] We used Stanford's Protégé-OWL tool to develop all taxonomies and ontologies. This is an open source, free ontology editor, available online at http://protege.stanford.edu/.

identification of functionality and associated resources that may be beneficial to users. In short, the metadata is evaluated to enable retrieval decisions, the software behavior representations enable searches based on functionality, and the ontologies point the user to helpful artifacts that they may not have initially considered.

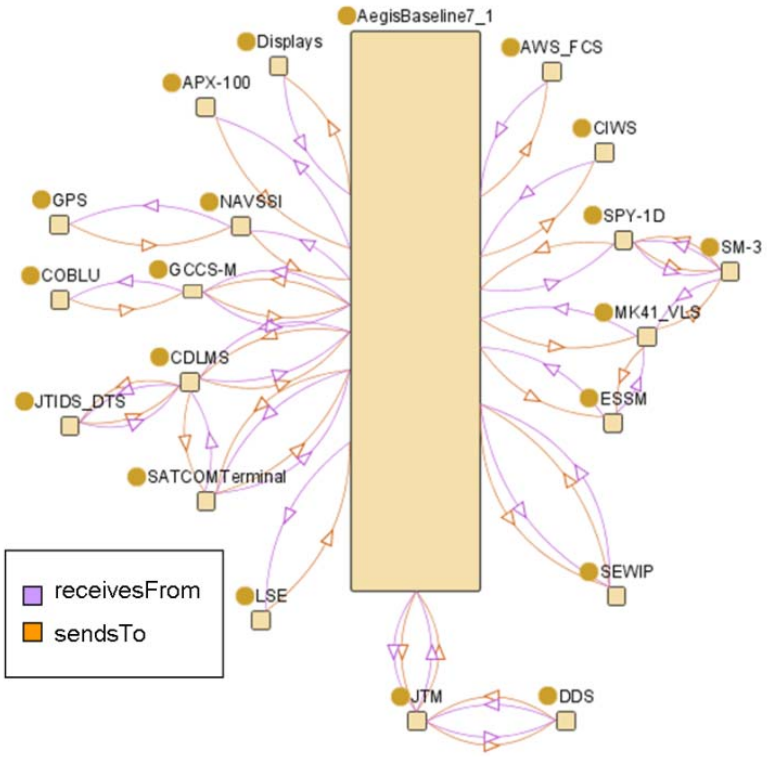


Figure 6. System Ontology Example (Aegis) (Jambalaya Graphic in Protégé)

Visualization

The tool must provide visualization techniques that will exploit the contextual information captured in the ontology and enable guided search capabilities. We suggest that multiple visualization tools be made available so that users can view the repository context and contents in a familiar setting.

One example of the type of tool that will be supported by the framework is a fish-eye graph. Fish-eye graphs display objects of interest to users, along with the relationships the objects have with other items. As the relationships interesting to users are explored, the graph highlights the item and brings it to the front of the display. Users can then weed out uninteresting items by removing from view the relationships that are not important. The results are a single or small grouping of items that users have found interesting with supporting information available by the click of a mouse.

Since domain information is captured in the repository framework, other architectural views may be used to present the repository contents in a display that is familiar to the user. For example, various views from the Department of Defense Architectural Framework (DoDAF) may be used as the backdrop to artifact nodes in order to provide a frame of

reference that is comfortable for users accustomed to the associated graphics. Artifact groupings may also be represented as UML diagrams to allow for easy interpretation by software developers.

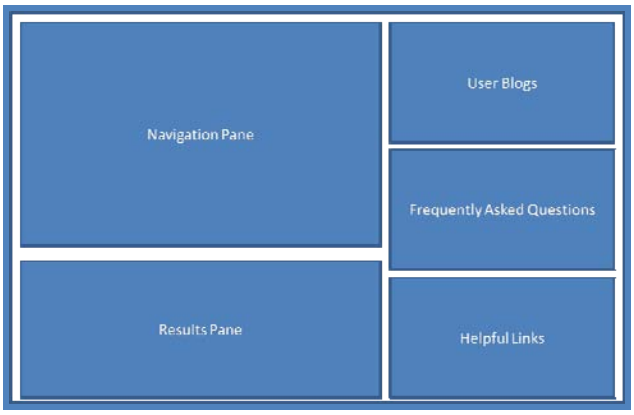


Figure 7. Screen Template

Use Case Demonstration

Here we describe, omitting some details, a use case scenario for the new tool to bring to light how each of the major features of the repository framework comes into play. This represents one possible scenario of interaction between a user and the repository system. The system reactions are described in terms of the individual windows on the screen that will update based on human-driven events. These windows include the Navigation Pane, Results Pane, User Blogs, Frequently Asked Questions, and Helpful Links, as shown in Figure 7.

In this scenario, the users need to build a replacement for a particular subsystem of the Aegis combat system, generically termed "Submodule B." They consult the SHARE repository to find artifacts that will help in the development of the requirements for the new subsystem. Potentially, there are requirements for an existing system that can be reused. There may also be additional artifacts to be discovered that may be helpful in the requirements development process. When the tool is first initiated, an initial (home) screen is provided to the user (Figure 8).

Figure 8. SHARE Home Screen

The initial navigation pane includes a short welcome and some guidance for how to get started. There is a list of initial questions that helps the system orient the view specifically for the user. These questions are intended to provide a starting point for the guided search by "anchoring" the initial search results appropriately within the ontologies based on relevant information provided by answering simple questions. If the user does not care to answer the questions, the guided search can begin immediately by pressing the SEARCH button. Tips are provided as popup windows that can be opened by left-clicking the hyperlink if the user is not sure how to get going, or if he is unsure about how to answer specific questions. The most often retrieved artifacts are presented as the default "results" in the result pane. Blogs that pertain to the repository system as a whole are presented (those with the most activity listed first) in the user blogs pane. The most often asked questions are provided, with a link to additional questions, in the FAQ pane. The questions displayed pertain to the entire repository. Finally, links to general information about the repository, related repositories identified by stakeholders, and other locations relevant to the content of the repository (Navy Open Architecture) are displayed in the helpful links pane.

The available answers in the drop down menus for each of the initial questions are dependent on the ontologies represented in the repository framework. As the questions are answered, each of the panes is updated to reflect each choice. A priority scheme is applied after each user selection, and items ranked highest according to the scheme may appear in the display if applicable. After a user has answered all questions in this scenario, the individual panes may appear as in Figure 9. The chosen answers appear in the drop down windows of the navigation pane. All other panes have been updated to reflect the choices made by the user to this point. The results, user blogs, FAQs, and helpful links panes all show reprioritized items that are associated with the requirements activity, the surface domain, the Aegis system, and Submodule B, where applicable. Additionally, items that are not specifically tagged to each of these selections may be listed based on the graphical distance captured through the use of the ontology relations.

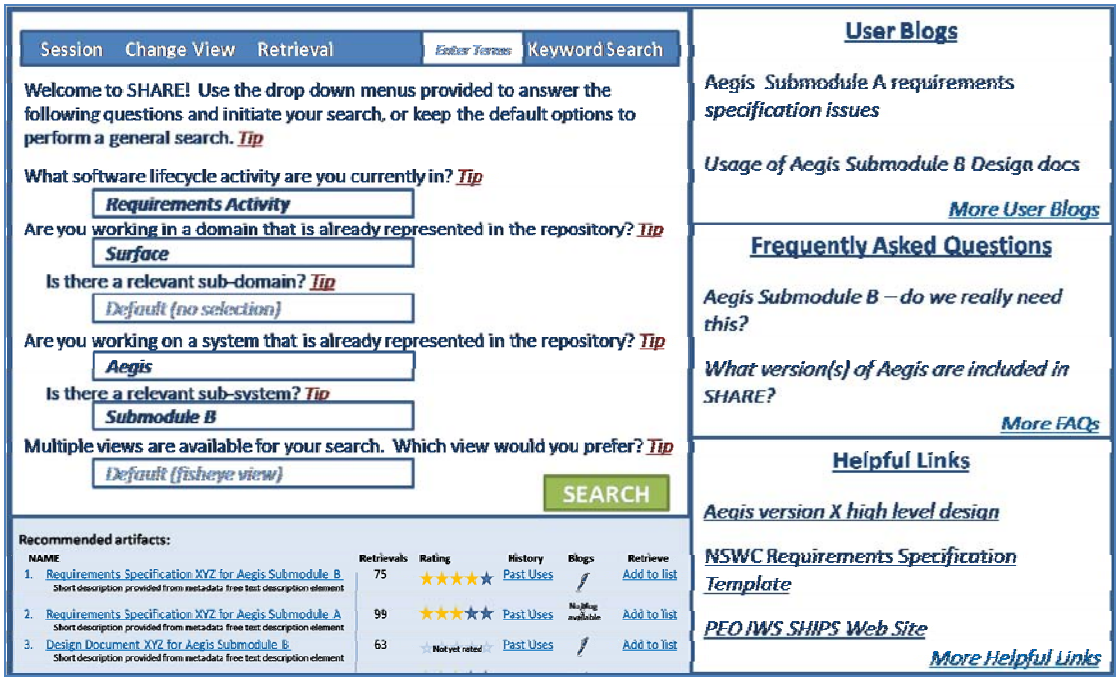


Figure 9. Home Screen After all Questions Answered

Once the user has selected the appropriate answers to each question, the SEARCH button is pressed. Since the default view was chosen, the navigation pane switches to the fisheye view of the repository contents (Figure 10). The fisheye view presents the artifacts of the repository as a graph that centers on the most relevant items. The positioning and size of the artifacts in the fisheye graph are determined by the prioritization scheme applied after certain user actions. The connectors between artifacts are the relations captured in the ontologies. Each of the types of relations is listed in an interactive menu that allows the user to turn the relations off and on depending on interest.

Additional features of the fisheye navigation include:

- Pop-up windows for artifacts and relations—Activated by mouse-scrolling actions, the artifact pop-up window contains a subset of the metadata for the artifact (see
-). The relation pop-up window describes how the two connected artifacts are related.
- Artifact detail page—Left-clicking on an artifact node opens an artifact detail page, which provides more of the artifact metadata.
- Action window—Right-clicking on an artifact node opens a drop-down action window that allows the user to open more information about the artifact or add it to the retrieval listing.

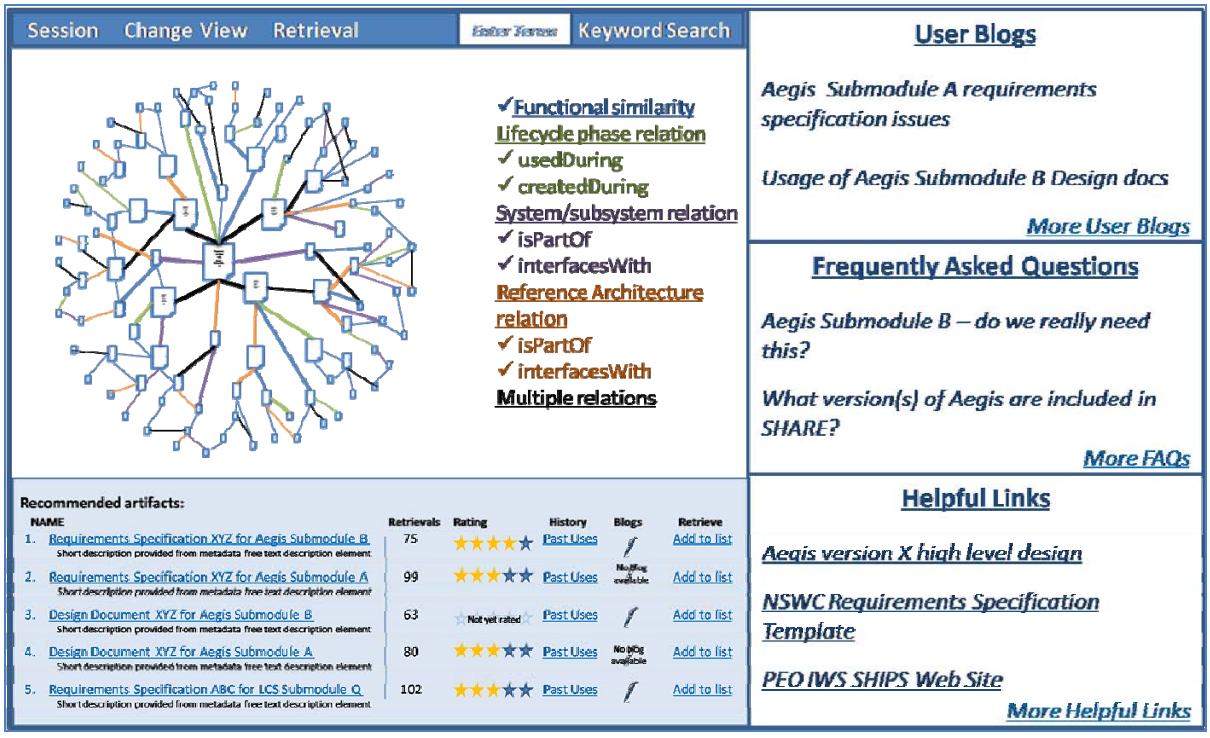


Figure 10. Initial Search Return—Fisheye View

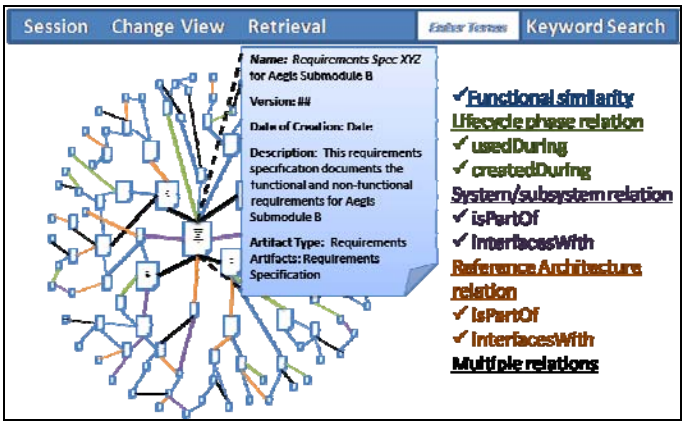


Figure 11. Artifact Pop-Up Window

Users can also add an item to their retrieval listing by selecting the option from the results pane. Each time the user selects an item for retrieval, the prioritization scheme is reapplied and each of the panes is updated to reflect the highest priority items.

After choosing all interesting items, the user selects Retrieval->Retrieve Items from the navigation pane drop down menu. A separate window is provided that contains the user's choices to this point, as shown in Figure 12. Select metadata is presented in addition to the names of artifacts to help the user review the list. The user can modify the list by deleting anything deemed irrelevant at this point. When the user is satisfied that the list of desired items is complete, the user presses the RETRIEVE button.

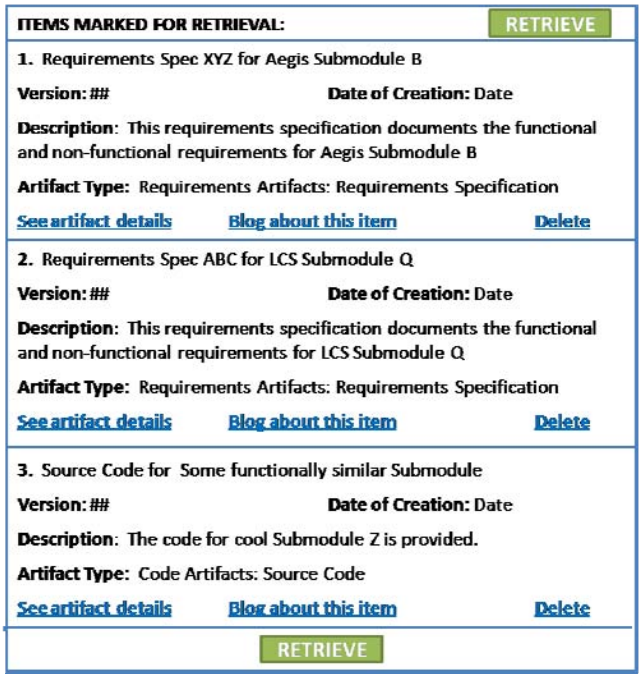


Figure 12. Retrieval List

Information is provided to assist the user in requesting and retrieving the items. Any items available for immediate download are made available using appropriate hyperlinks. Items that require request/approval processes are also enabled through a step-by-step automated process. This extra step is required for repositories that have security and

intellectual property limitations, such as the SHARE repository. If the Ontology Based Software Reuse Repository System is developed for an open source repository, it should be possible to simply provide links to the artifacts themselves. In this case, it would be desirable to replace the "Retrieve" column in the results pane with a "Download" column, and to add a menu option for downloading the artifact when the user right-clicks on an item in the navigation pane.

Related Work

Sugumaran and Storey propose the use of domain knowledge in repositories to aid in the natural language processing of queries for component retrieval (Sugumaran & Storey, 2003). In their prototype system, the ontology captures synonyms and relations between objects in the domain. The system enables the user to enter queries using natural language, and the ontology enables more coverage in the returned items by including items that contain the relations captured in the ontology.

This work is closely related to the system proposed herein, since they both address some limitations of traditional keyword and faceted classification-based searches. However, there are several key differences. First, the Sugumaran et al. ontology is limited to a single view of the typical objects and terms within a specific application domain; whereas our approach includes multiple views as described. Second, the visualization enabled by the ontology is vastly different. In the Sugumaran et al. approach, syntactic analysis is conducted on a query entered through a free text interface, resulting in lists of processes, actions, and matching or related components that the user can then choose to view in more detail. Our approach enables the user to navigate the repository contents in a more interactive way. Finally, the use of the ontology to provide a lexicon for matching terms is extended in our approach since artifacts in the repository are captured as individual items in the ontology classes. This approach provides wider use of the ontology in representation of repository contents and user interaction.

Yao and Etzkom also focus on the use of ontologies for enhancing search retrieval based on natural language queries. They extend the idea by suggesting the use of Semantic Web technologies such as RDFS/DAML+OIL to apply the methodology to the World Wide Web as a large software repository (Yao & Etzkorn, 2004).

Summary and Future Work

In this paper, we have presented an ontology-based approach for the development of a software reuse repository. Our claim is that the knowledge captured by the ontologies enables new ways of discovering desired software artifacts based on computer aided navigation rather than the more traditional query/response discovery. We described the repository framework that provides the contextual depth to support such navigation and demonstrated the approach using a use case.

Throughout the project we have identified several areas for future work. First, we recognize a need to investigate the automated population of artifact metadata. A significant challenge will be the generation of XML metadata from existing reusable resources and help for users in describing future submissions to the repository. Current approaches for automatic generation of metadata from content libraries should be explored for potential application to the ontology-based repository and more specifically for the SHARE metadata context. We suspect that a combination of techniques will be useful.

Second, providing a practical software behavior representation remains a challenging area for continued exploration. The current work provides an identification of principal functionality of an artifact through the CSFL and possible description of operations and input/output messages from a service perspective using WSDL. The work is admittedly preliminary. Research into related areas of Semantic Web Services, Business Process Execution Language, and others continues to hold promise for this aspect of the repository framework.

Third, additional user views may be desirable other than those we have described here. Some investigation into the feasibility of translating the ontological information between various model types is warranted.

Finally, in addition to the Navy's CSFL, similar lists have been developed for operational activities (COAL) and for information elements (CIEL). It would be interesting to express these taxonomies in OWL, as was done with CSFL, and then to create interrelationships across the classes, for example, to determine what information elements are generally employed in performing certain system functions, or what information elements are generally produced by performing certain system functions. Further exploration with subject matter experts (SMEs) is needed to determine potential benefit from such approaches.

Acknowledgments

This research would not have been possible without the financial and moral support of Mr. Nick Guertin, Director of the Navy Future Combat Systems Open Architecture Program. I would also like to sincerely thank Dr. Mikhail Auguston, who has been my mentor and guide throughout this research.

References

Guo, J., & Luqi. (2000). A survey of software reuse repositories. In *Proceedings of the Seventh IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, 2000 (ECBS 2000) (pp. 92-100).

Hassan, A. E. (2008). The road ahead for mining software repositories. In *Proceedings of the 2008 IEEE Frontiers of Software Maintenance* (pp. 48-57). Beijing, China.

Johnson, J., & Blais, C. (2008). *Software hardware asset reuse enterprise (SHARE) framework: related work and development plan*. Monterey, CA: Naval Postgraduate School.

Johnson, J., & Blais, C. (2008). *Software hardware asset reuse enterprise (SHARE) repository framework final report: Component specification and ontology*. Monterey, CA: Naval Postgraduate School.

Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software*, *12*(6), 42-50.

Object Management Group. (2005). *Reusable asset specification* (Vers. 2.2).

Shiva, S., & Shala, L. (2007). Software reuse: Research and practice. In *Proceedings of the Fourth International Conference on Information Technology, 2007 (ITNG '07)* (pp. 603-609).

Sugumaran, V., & Storey, V. C. (2003). A semantic-based approach to component retrieval. *The DATA BASE for Advances in Information Systems*, *34*(3), 8-24.

Yao, H., & Etzkorn, L. (2004). Towards a semantic-based approach for software reusable component classification and retrieval. In *Proceedings of the ACM 42nd Southeast Conference (ACMSE '04)* (pp. 110-115). Hunstville, AL.

THIS PAGE INTENTIONALLY LEFT BLANK

2003 - 2010 Sponsored Research Topics

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case
- Budget Scoring
- Budgeting for Capabilities-based Planning

- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-tem Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)
- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)

- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

THIS PAGE INTENTIONALLY LEFT BLANK

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.org

Acquisition Research Program:
Creating Synergy for Informed Change

Ontology-based Software Reuse Repository

Jean Johnson

Lecturer, Naval Postgraduate School

Topics

- Background
- Research Goals
- Repository Tool
  - Guided Search
  - Repository Framework
  - Visualization
- Use Case Demonstration

Background

- Need for improvements
  - Software repository capabilities often sighted as inhibitor to software reuse
  - Robust search and discovery methods needed
- SHARE Repository
  - Enables reuse of combat systems software and related assets
  - Est. August 2006 by PEO IWS
  - NPS researchers tasked to consider areas of improvement for the repository

Research Goal

- Improved software repository capabilities
- Areas of improvement:
  - Organization of repository contents
    - Robust metadata
    - Include all relevant artifact types
    - Ontologies to capture context for repository contents
  - Search and discovery methods
    - Guided search
    - Multiple Views

Guided Search

- Smart navigation of repository contents vs. typical search and return
- Point and click interface
- User context incorporated into search process
- Results are continuously reprioritized based on user actions

Repository Framework

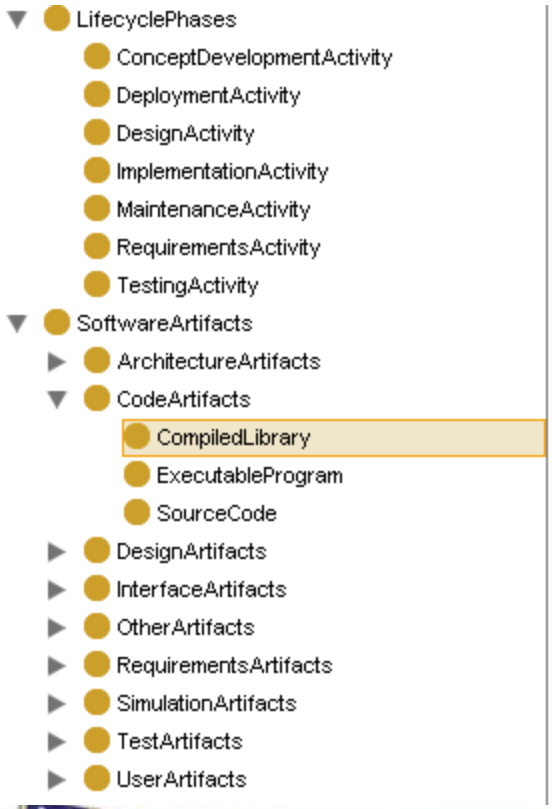
- Component Specification - a description or model of the items in the repository
  - "Typical" Metadata - information about an asset/artifact
  - Software Behavior Description – a searchable representation of the software asset's behavior
- Ontology – a contextual model of the repository items describing their relationships to aid in associating artifacts with user needs

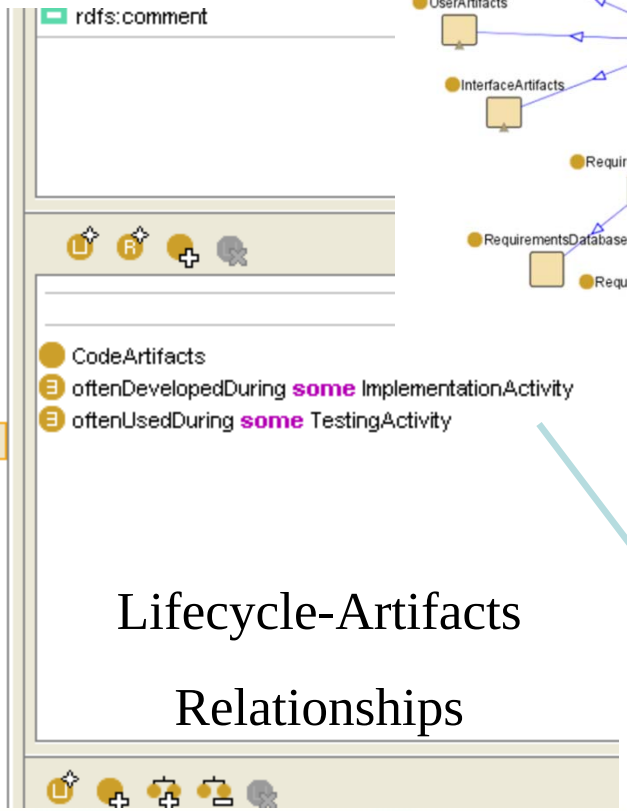
Repository Framework - Ontologies

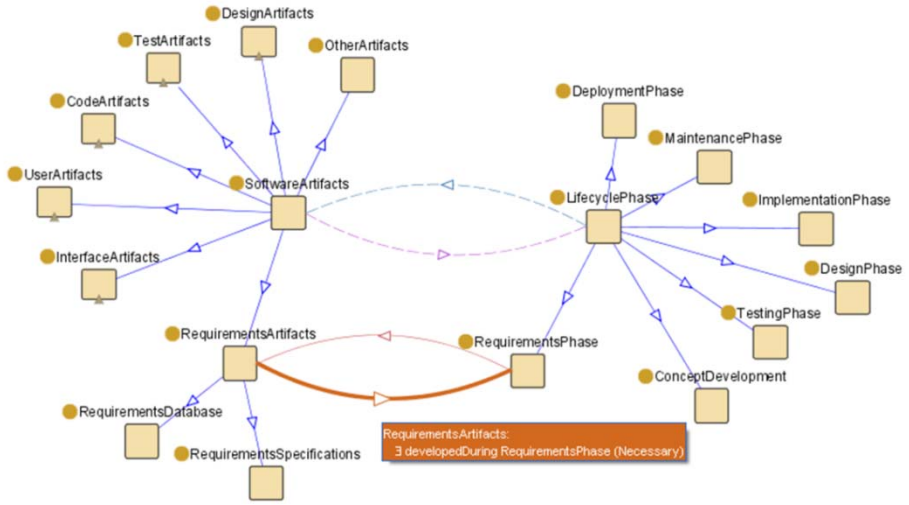
- Multiple sources of context for repository artifacts
  - Artifact's place in the Software Engineering Lifecycle
  - System Architectures (Aegis, SSDS, etc.)
  - Surface Navy Open Architecture reference architecture
- OWL-DL (Description logic) representations included in previous reports

Lifecycle-Artifact Ontology

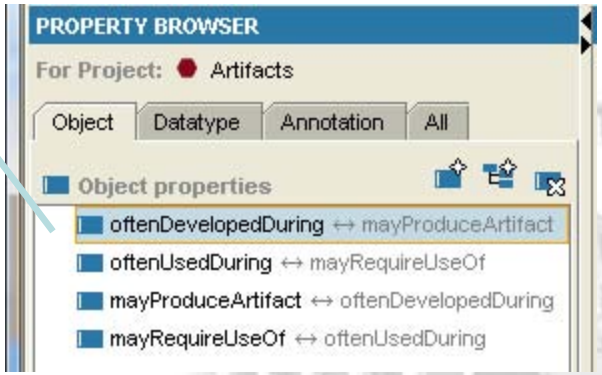
- Links artifact types to development activities





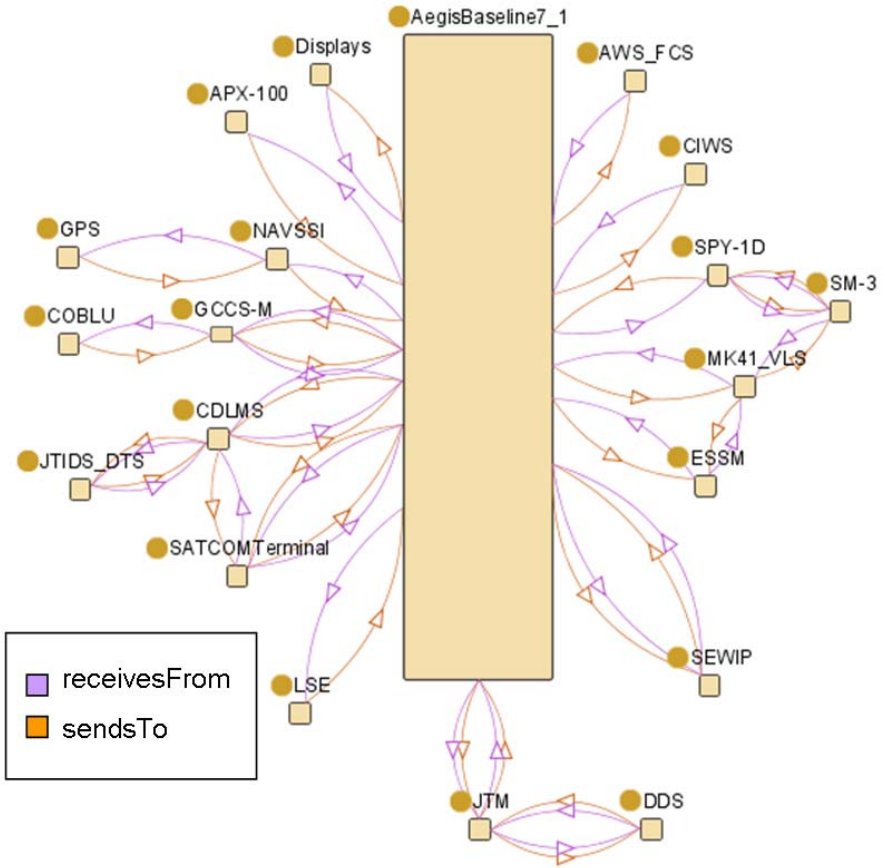


Lifecycle-Artifacts Relationships



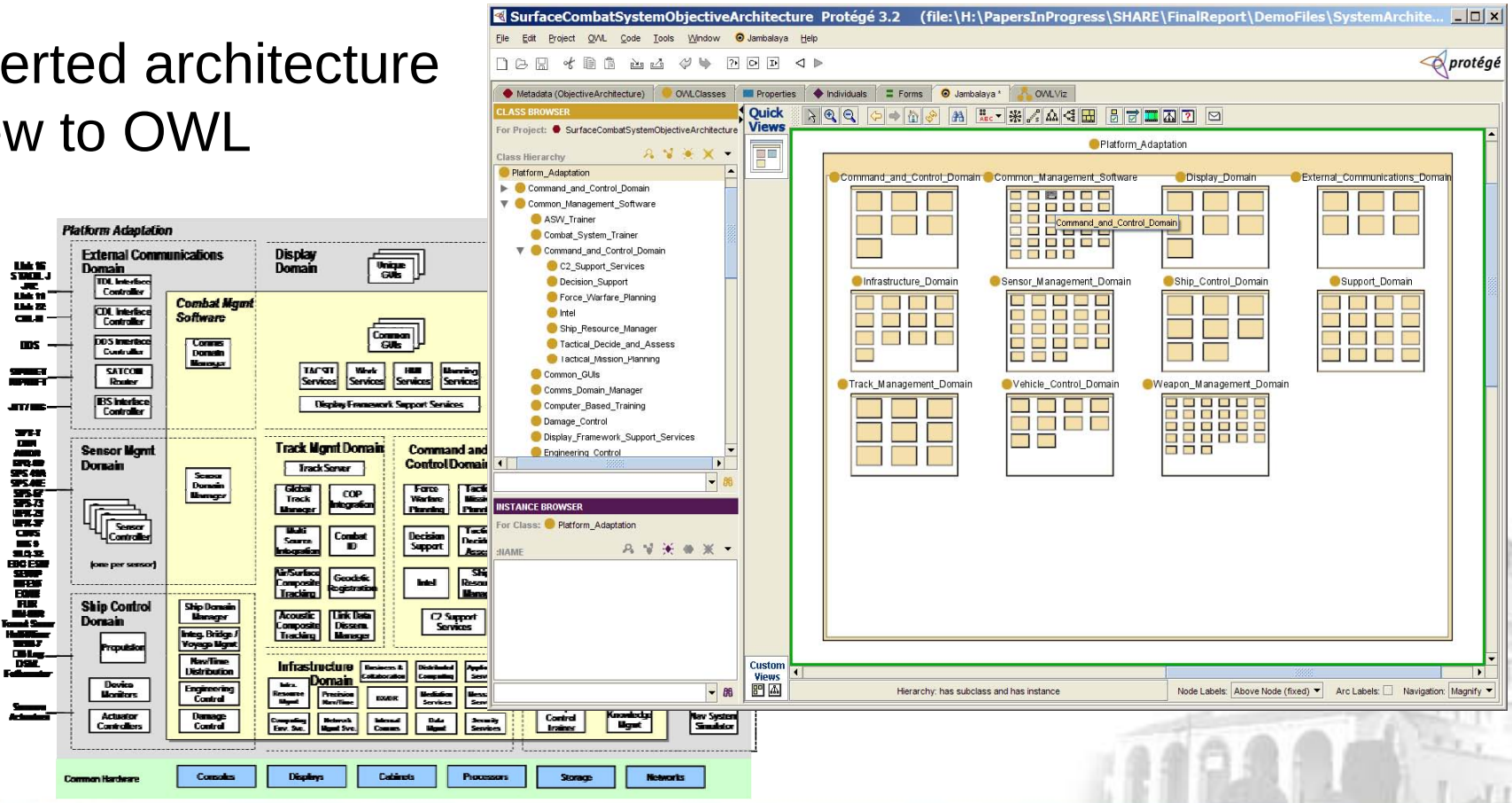
System Architecture

- Captures
  - System-subsystem relationships
  - Interfaces
  - Any other desired architectural relationships
- Example from Aegis SV-1 available in RDA CHENG Naval Architecture Repository System (NARS)



Surface Combat System Top Level Objective Architecture

Converted architecture view to OWL



Visualization

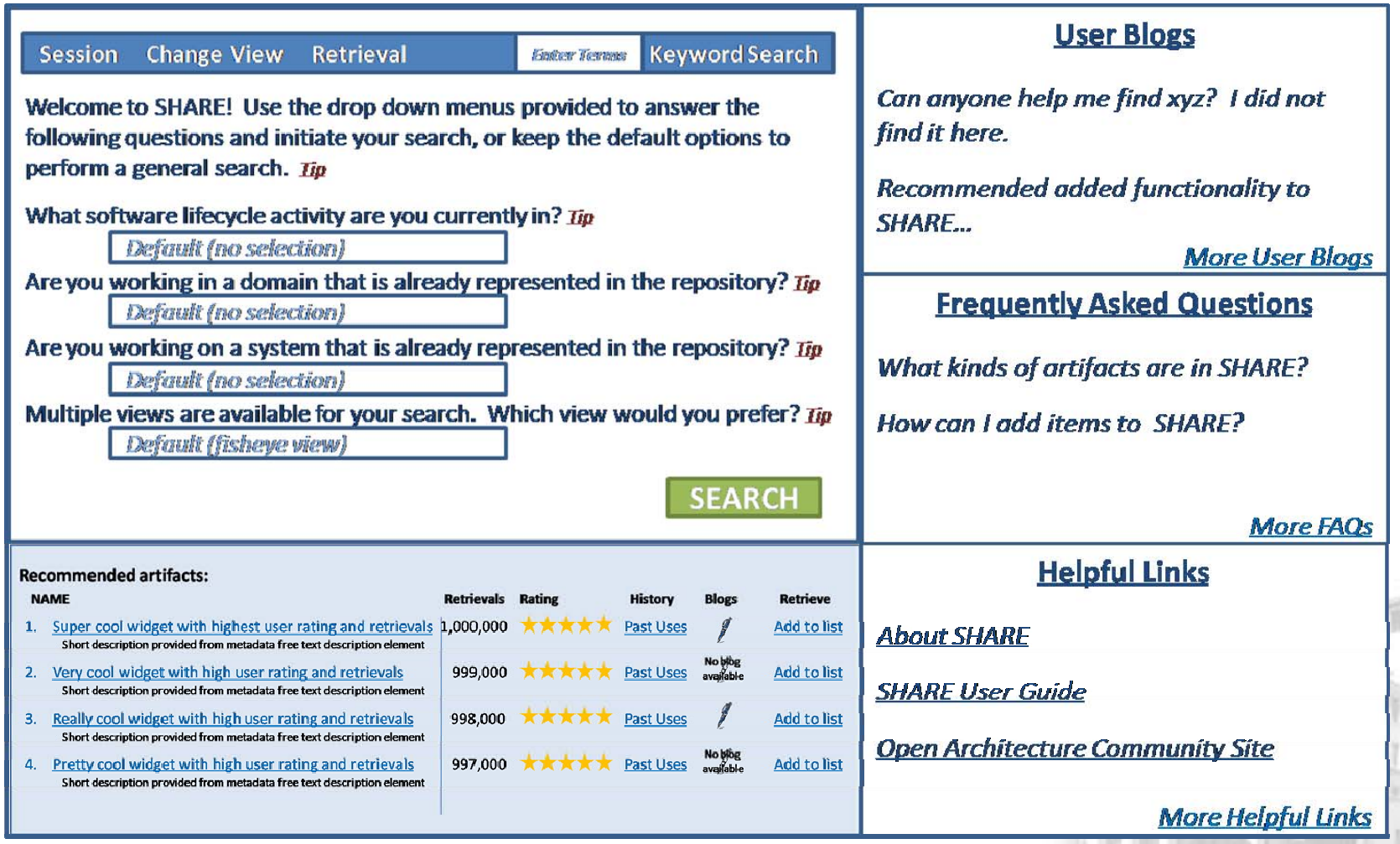
- Multiple visualization tools allow users to explore contents in a comfortable setting.
- Different types of views suggested:
  - Fisheye Graph (Use Case shows)
  - DoDAF Views
  - UML Representations

Use Case Demonstration

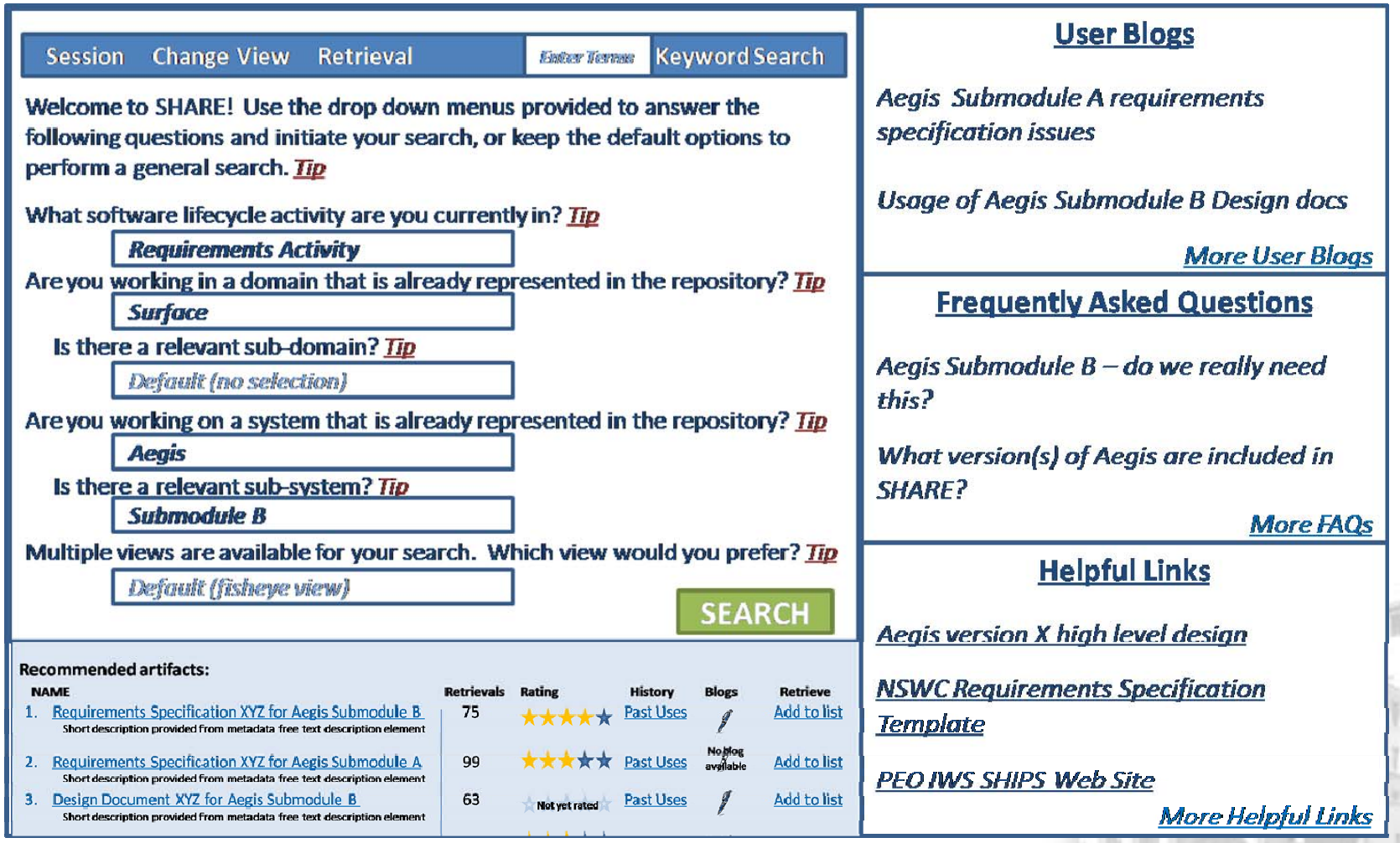
Overall Description

- Replacement for a subsystem of Aegis, generically termed “Submodule B”.
- SHARE repository consulted to find artifacts that will help in the development of the new subsystem ***requirements.***
- Potentially, there are existing system requirements that can be reused.
- There may also be additional artifacts to be discovered that may be helpful.

Home Screen

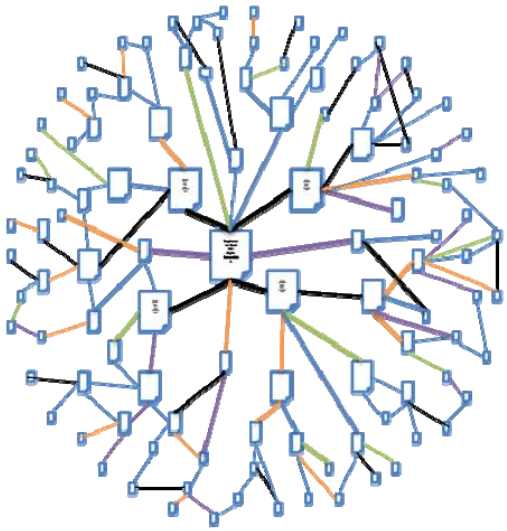

Session
Change View
Retrieval
Enter Terms
Keyword Search
Welcome to SHARE! Use the drop down menus provided to answer the following questions and initiate your search, or keep the default options to perform a general search. Tip
What software lifecycle activity are you currently in? Tip
Default (no selection)
Are you working in a domain that is already represented in the repository? Tip
Default (no selection)
Are you working on a system that is already represented in the repository? Tip
Default (no selection)
Multiple views are available for your search. Which view would you prefer? Tip
Default (fisheye view)
SEARCH
Recommended artifacts:
NAME
Retrievals
Rating
History
Blogs
Retrieve
1. Super cool widget with highest user rating and retrievals 1,000,000 Past Uses Add to list
Short description provided from metadata free text description element
2. Very cool widget with high user rating and retrievals 999,000 Past Uses No blog available Add to list
Short description provided from metadata free text description element
3. Really cool widget with high user rating and retrievals 998,000 Past Uses Add to list
Short description provided from metadata free text description element
4. Pretty cool widget with high user rating and retrievals 997,000 Past Uses No blog available Add to list
Short description provided from metadata free text description element
User Blogs
Can anyone help me find xyz? I did not find it here.
Recommended added functionality to SHARE...
More User Blogs
Frequently Asked Questions
What kinds of artifacts are in SHARE?
How can I add items to SHARE?
More FAQs
Helpful Links
About SHARE
SHARE User Guide
Open Architecture Community Site
More Helpful Links

Initial Questions Answered



Fisheye Navigation – Initial return

Session Change View Retrieval | Enter Terms | Keyword Search



✓Functional similarity
Lifecycle phase relation
✓ usedDuring
✓ createdDuring
System/subsystem relation
✓ isPartOf
✓ interfacesWith
Reference Architecture relation
✓ isPartOf
✓ interfacesWith
Multiple relations

Recommended artifacts:

	NAME	Retrievals	Rating	History	Blogs	Retrieve
1.	Requirements Specification XYZ for Aegis Submodule B Short description provided from metadata free text description element	75	★★★★★	Past Uses		Add to list
2.	Requirements Specification XYZ for Aegis Submodule A Short description provided from metadata free text description element	99	★★★★★	Past Uses	No blog available	Add to list
3.	Design Document XYZ for Aegis Submodule B Short description provided from metadata free text description element	63	Not yet rated	Past Uses		Add to list
4.	Design Document XYZ for Aegis Submodule A Short description provided from metadata free text description element	80	★★★★★	Past Uses	No blog available	Add to list
5.	Requirements Specification ABC for LCS Submodule Q Short description provided from metadata free text description element	102	★★★★★	Past Uses		Add to list

User Blogs

Aegis Submodule A requirements specification issues

Usage of Aegis Submodule B Design docs

More User Blogs

Frequently Asked Questions

Aegis Submodule B – do we really need this?

What version(s) of Aegis are included in SHARE?

More FAQs

Helpful Links

Aegis version X high level design

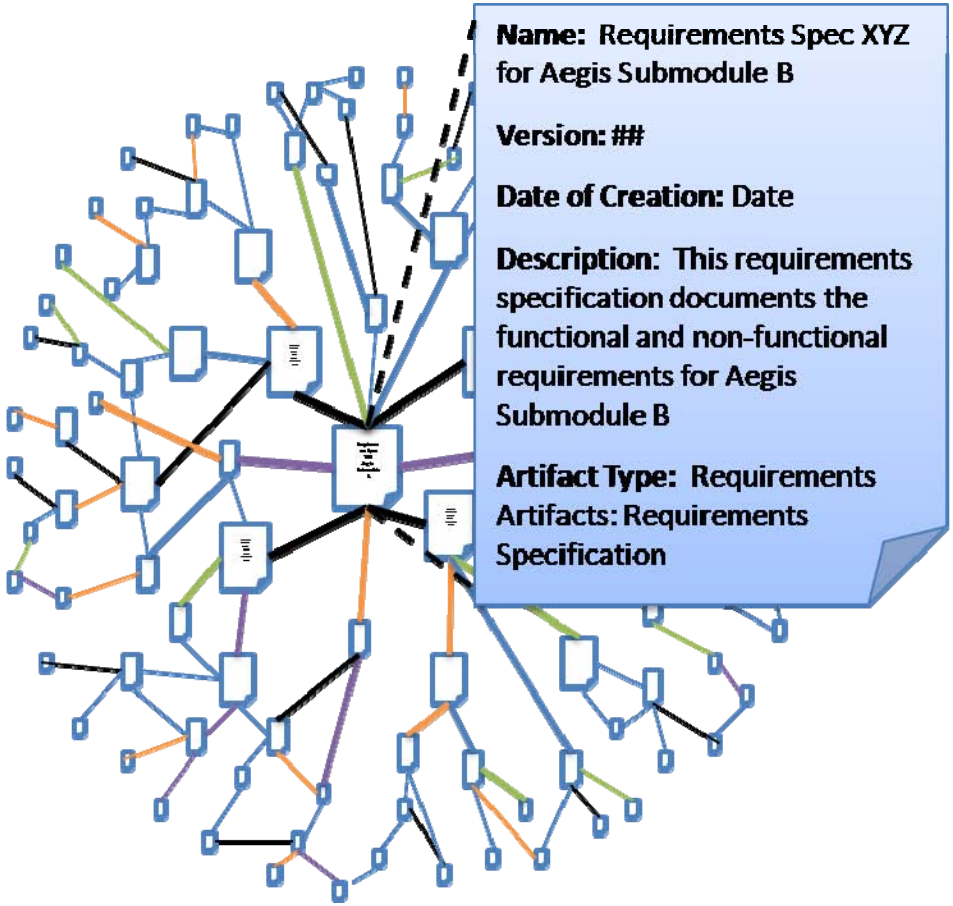
NSWC Requirements Specification Template

PEO IWS SHIPS Web Site

More Helpful Links

Artifact Details

Session Change View Retrieval | Enter Terms | Keyword Search



Name: Requirements Spec XYZ for Aegis Submodule B

Version: ##

Date of Creation: Date

Description: This requirements specification documents the functional and non-functional requirements for Aegis Submodule B

Artifact Type: Requirements Artifacts: Requirements Specification

✓Functional similarity

Lifecycle phase relation

✓ usedDuring

✓ createdDuring

System/subsystem relation

✓ isPartOf

✓ interfacesWith

Reference Architecture relation

✓ isPartOf

✓ interfacesWith

Multiple relations

Artifact Detail Page

Name: Requirements Spec XYZ for Aegis Submodule B

Version: ## **Date of Creation:** Date

Description: This requirements specification documents the functional and non-functional requirements for Aegis Submodule B

Artifact Type: Requirements Artifacts: Requirements Specification

Retrievals	Rating	History	Blogs	Retrieve
75	★★★★★	Past Uses		Add to list

Contribution Rationale: Submitter rationale for why it is included in the repository (free text)

Applicable Systems: Aegis: Submodule B

Objective Architecture Tags: Command and Control: Global Track Manager

Software Behavior Description: CSFL Function 1, CSFL Function 2

History:

Development Status: Complete

Pedigree:

Newer Version Of: Aegis Submodule B version XX

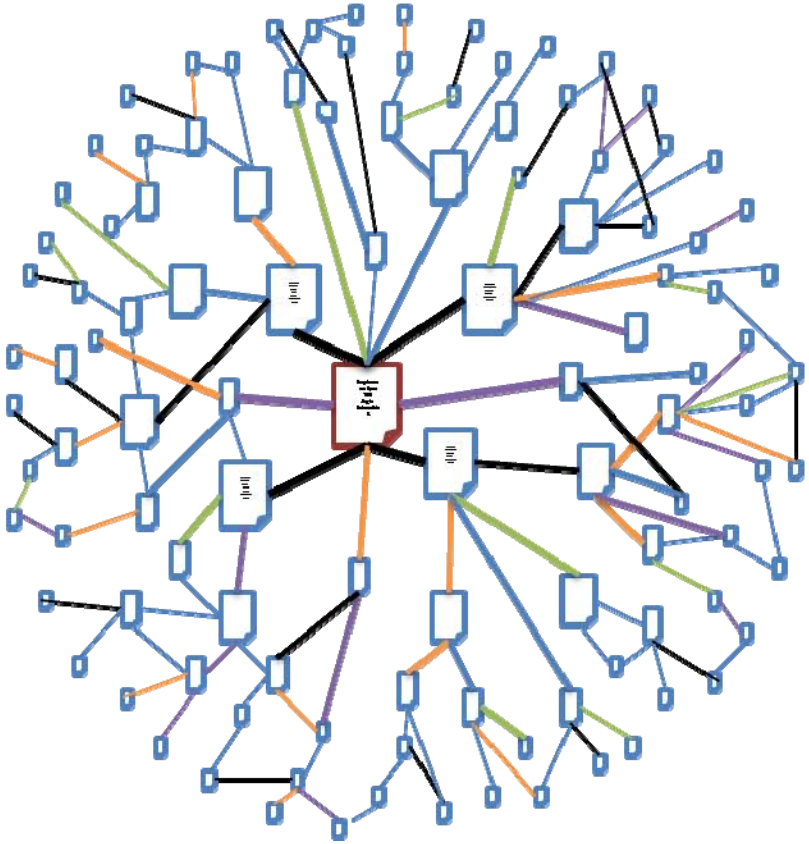
Interdependencies:

Interfaces With: Aegis Submodule A, Aegis Submodule C

Users who retrieved this item also retrieved:
List of applicable artifacts with hyperlinks to the relevant artifact detail page.

Retrieved item highlighted

Session Change View Retrieval | Enter Terms | Keyword Search



✓Functional similarity

Lifecycle phase relation

✓ usedDuring

✓ createdDuring

System/subsystem relation

✓ isPartOf

✓ interfacesWith

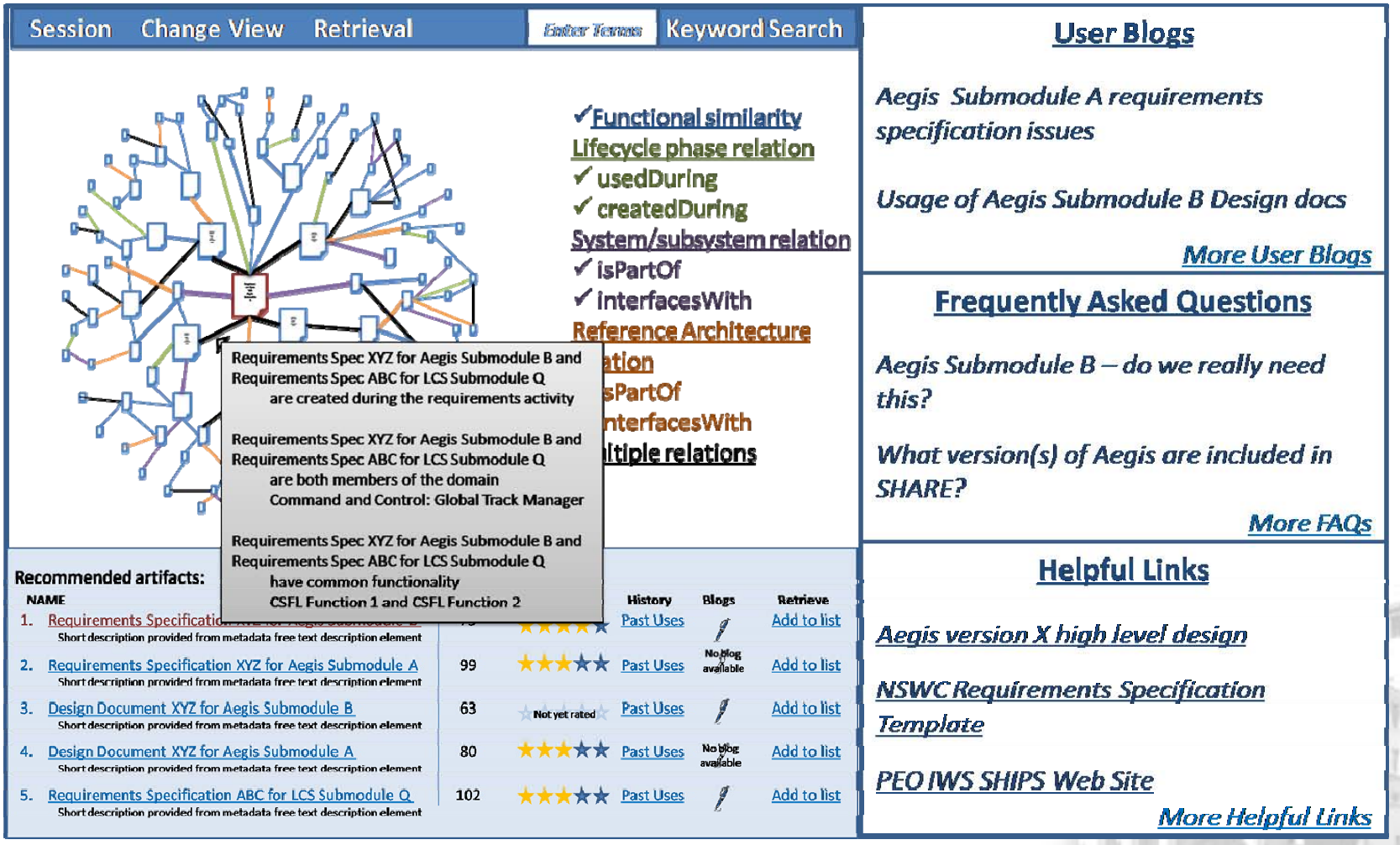
Reference Architecture relation

✓ isPartOf

✓ interfacesWith

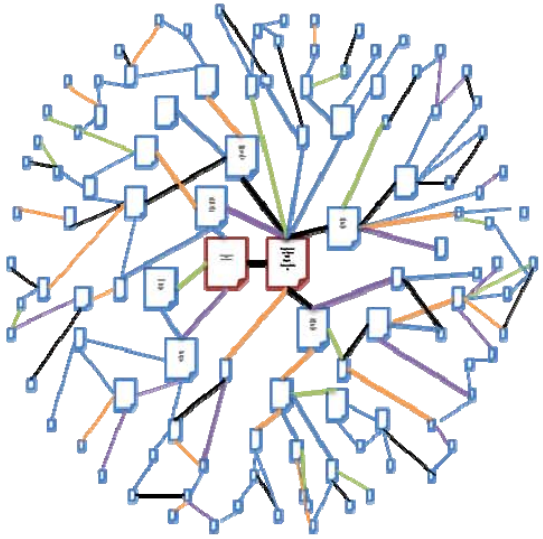
Multiple relations

Relation Popup Window



Pane updates based on user action

Session | Change View | Retrieval | Enter Terms | Keyword Search



✓Functional similarity
Lifecycle phase relation
✓ usedDuring
✓ createdDuring
System/subsystem relation
✓ isPartOf
✓ interfacesWith
Reference Architecture relation
✓ isPartOf
✓ interfacesWith
Multiple relations

Recommended artifacts:

	NAME	Retrievals	Rating	History	Blogs	Retrieve
1.	Requirements Specification XYZ for Aegis Submodule B Short description provided from metadata free text description element	75	★★★★★	Past Uses		Add to list
2.	Requirements Specification ABC for LCS Submodule Q Short description provided from metadata free text description element	102	★★★★★	Past Uses		Add to list
3.	Design Document XYZ for Aegis Submodule B Short description provided from metadata free text description element	63	Not yet rated	Past Uses		Add to list
4.	Design Document ABC for LCS Submodule Q Short description provided from metadata free text description element	65	★★★★★	Past Uses	No blog available	Add to list
5.	Requirements Specification 123 for SSDS Short description provided from metadata free text description element	55	★★★★★	Past Uses		Add to list

User Blogs

Aegis Submodule A requirements specification issues

LCS Submodule Q requirements

More User Blogs

Frequently Asked Questions

Aegis Submodule B – do we really need this?

How do I find details about the objective architecture developed for surface?

More FAQs

Helpful Links

PEO IWS SHIPS Web Site

NSWC Requirements Specification Template

Command and Control Best Practices

More Helpful Links

Retrieval List

ITEMS MARKED FOR RETRIEVAL: RETRIEVE

1. Requirements Spec XYZ for Aegis Submodule B

Version: ## **Date of Creation:** Date

Description: This requirements specification documents the functional and non-functional requirements for Aegis Submodule B

Artifact Type: Requirements Artifacts: Requirements Specification

See artifact details Blog about this item Delete

2. Requirements Spec ABC for LCS Submodule Q

Version: ## **Date of Creation:** Date

Description: This requirements specification documents the functional and non-functional requirements for LCS Submodule Q

Artifact Type: Requirements Artifacts: Requirements Specification

See artifact details Blog about this item Delete

3. Source Code for Some functionally similar Submodule

Version: ## **Date of Creation:** Date

Description: The code for cool Submodule Z is provided.

Artifact Type: Code Artifacts: Source Code

See artifact details Blog about this item Delete

RETRIEVE

Summary

- Ontology-based approach for organizing a software reuse repository
- Computer aided navigation techniques are enabled based on relationships in ontologies
- Demonstrated by Use Case scenario

Questions?

Jean Johnson
Systems Engineering Dept.
Naval Postgraduate School
jmjohnso@nps.edu
(757)574-7563