INFORMATION AWARENESS OFFICE

Scientia Est Potentia

DARPA

Total Information Awareness Program (TIA)
System Description Document (SDD)

Version 1.1*

July 19, 2002

* See Appendix A: Document Revision History

Gregory Mack, PhD (mackg@saic.com), System Architect
Hicks and Associates, Inc.

Business Model
B. Bebee (bebeeb@saic.com)
I. Shafi

Hicks and Associates, Inc.

Systems Model
G. Wenzel (wenzel_greg@bah.com)
B. Medairy
E. Yuan
Booz-Allen and Hamilton, Inc.

1. Executive Summary

The Total Information Awareness (TIA) Program seeks to provide a true end-to-end capability for analysts and decision-makers. As a result, the TIA System development effort is characterized by four key elements:

1. **Innovative architecture** – based on the Open Grid Service Architecture (OGSA) concepts, designed to be flexible, robust and scalable
2. **A rapid turn-around experimentation process** – using real data and addressing real operational needs
3. **A technology refresh process** – enabling the introduction of the latest technologies in real operational settings
4. **An infrastructure development and tool hardening process** – facilitating full transition of the best of breed tools to users in their operational environments.

The TIA System Description Document (SDD) provides insight into each of the elements mentioned above. A constantly evolving document, it provides "Topsight," using the Unified Modeling Language (UML) to visually tie all levels of the system together, generating insight from architectural patterns, and facilitating drill-down to understand underlying rationale. The SDD is designed for three primary groups of people:

1. **Domain Users (analysts and decision-makers) –** mapping their functional needs to tools in the form of Use Case Models.
2. **Developers (system and tool)** – classifying and documenting core services and user interfaces (functional requirements) in the form of Class and Collaboration Diagrams, as well as links to utilities helpful in integrating applications with the services.
3. **Experiment Designers** – identifying the system infrastructure used by the analyst's applications and the customer's platform.

Each class of user has their own entry point to the architecture and a path to navigate to any other element of the SDD. The TIA System Architecture is comprised of three interdependent models, each with a corresponding set of views.

1. **Core Conceptual Model (Section 2)** – creates a set of consistent unifying principles to link the Business Model and the System Model.
2. **Business Model (Section 3.1)** – captures the TIA functionality from the domain perspective.
3. **Systems Model (Section 3.2)** – captures the design of the supporting IT system.

The division into Business and Systems Models provides benefits in three areas:

1. Improving the specification of the user needs by clearly distinguishing between functional (business) and system requirements.

2. Facilitating multiple implementations of the processes for service composition.
3. Providing a foundation for evaluation of the system at the process level.

1.1. Core Conceptual Model

The Core Conceptual Model has a Core Model and an Application Model. The Core Model (See Section 2.1.1) provides the conceptual underpinnings for operations, business and system. This becomes an important aspect in the practical inclusion of the semantic aspects of the services and content. The Application Model (See Section 2.2) provides the conceptual underpinnings for the design of applications.

1.2. Business Model

The Business Model is a description, from a domain perspective, of the tasks and activities, operational elements, metrics, and information flows required to accomplish or support operations. The Business Model contains five views: (1) Actor Roles View (See Section 3.1.1), (2) Business Process View (See Section 3.1.2), (3) Business Information View (See Section 3.1.4), (4) Business Application Service View (See Section 3.1.3), and (5) Experiment Deployment View (See Section 3.1.5).

1.3. System Model

The System Model describes the arrangement, interaction, and interdependence of system elements at a systems and platform level. It provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established. The System Model contains seven views, the first five are system analogs of the Business Model views, with a Component Catalog View (See Section 3.2.8) and a Core Services View (See Section 3.2.5) added. The Component Catalog view enables the user to easily identify the source of a component (vendor and funding program).

1.4. Distinguishing Aspects of the Approach

The TIA SDD has several additional distinguishing aspects. First, there is an Experiment Deployment View (See Section 3.1.5) for each experiment. It is a *snapshot of the TIA System*, as built, at that point in time. It serves not only as the documentation for the experimental objectives and setup, but also as a chronology, which constitutes an experimental configuration management system that could be used to roll back and repeat an experiment. The rest of the Business Model constitutes the "as envisioned" model. Therefore, the chronology also provides a retrospective road map of the evolution of the TIA System. In addition, the chronology helps TIA Program Management map the progress of the system into Program goals. The System Implementation View (See Section 3.2.9) provides the same capability for the system infrastructure evolution. Second, it contains a *Semantic Architecture* as part of the Information View (See Section 3.2.6). A working definition of semantics is provided that facilitates the definition and implementation of *semantic interfaces* for services. The Semantic Architecture is comprised of a means for accessing content semantics through services and a mechanism for communicating service semantics along with the content.

Table of Contents

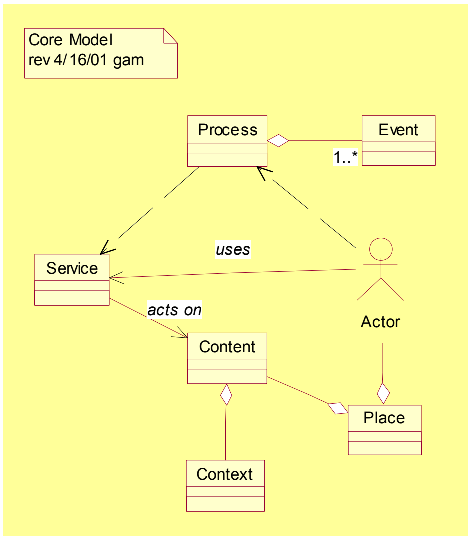
2. Rationale and an Overview (Core Conceptual Model)

2.1. Topsight

"Topsight" is the ability to "see the whole thing" — and to plunge in and explore the details. [Gelernter, Mirror Worlds, 1991]

2.1.1. The TIA System has been designed from its inception to operate within complex situations on a huge scale.

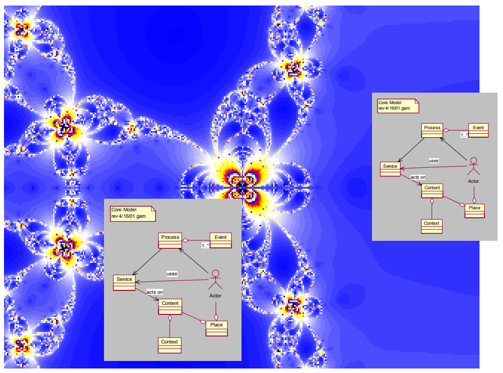
- There are seven entities in the Core Model
  - **Processes** cause **Actors** and **Services** to act on **Content** in **Places**.
  - **Actors** can be humans or "bots."
  - **Places** are virtual.
  - **Processes** may include both systems and business processes.
  - In the main, the system is **event-driven, which** can be either system or external.
  - All entities have **Context**.



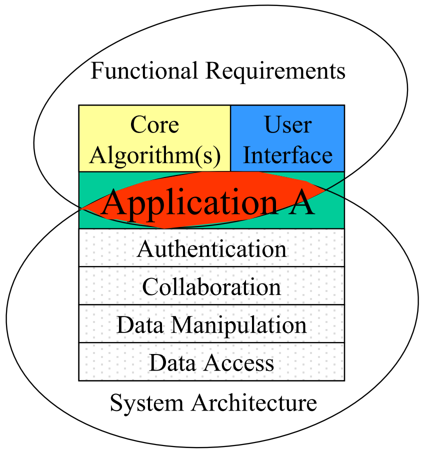
Each of the entities have metadata, even the context itself...

2.1.2. A critical design element in these situations is containing complexity. Our approach uses the fractal principal of self-similarity.

- A self-similar system "looks" the same when viewed at any scale.
- Keys to a successful Self-Similar Architecture:
  - "Instrumentation"
    - Ability/responsibility for each node to expose descriptive information to other nodes.
  - Role-based interfaces.
  - Comprehensive Context with dynamic schemas.
  - Information currency (i.e., *CIPs*).



2.1.3. The TIA System Description provides "Topsight" to the TIA System, tying all levels of the system together and facilitating drill-down.

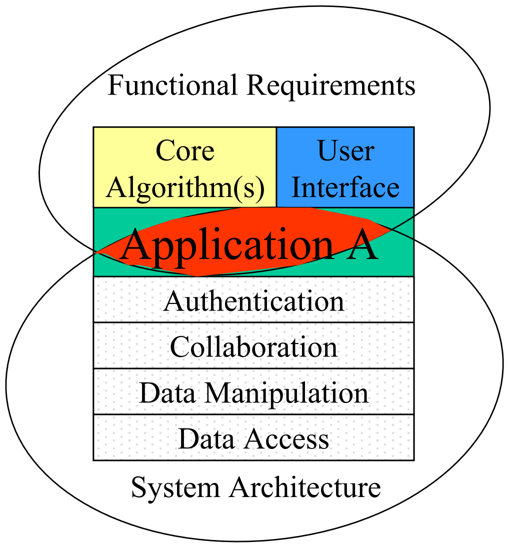


- It provides a *reference model* for:
  - **Domain users**
    - It maps the functional requirements from the Applications to the Services.
  - **System developers**.
    - It classifies and documents core algorithms and user interfaces (functional requirements) as UML Use Case Models.
      - Provide, where appropriate, SDKs to application developers for integrating with the services
      - Link to documentation and experiments
  - **Experiment designers**.
    - It identifies the system infrastructure used by the applications as developed in UML Diagrams.
      - Identify, describe, and define the services required by the application
      - Identify, describe, and define the services required by the customer's platform (IAC)
- A web-based (XML) *system design document* provides access to users (IACs, DARPA PMs, Contractors).

2.2. TIA Application

2.2.1. The Application is where Function meets the System.

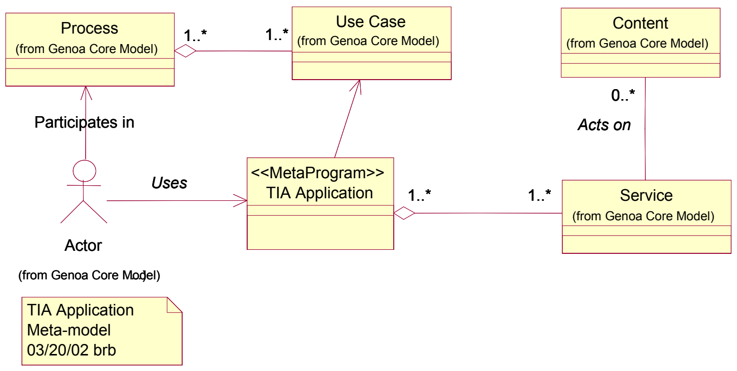
- Users care about how an application helps them do their job.
  - A specification of the core algorithms and the user interface
- The System "cares" about the services and infrastructure required to deploy functional capabilities to the users.
- An Application will capture a specification of a (set of) functional requirement (s) in a *Semantic Wrapper* (i.e., the self-describing nature of XML documents)



Eventually, the specification will become executable.

2.2.2. The definition of a TIA Application is derived from the Core Model and links Processes, Use Cases, and Services.

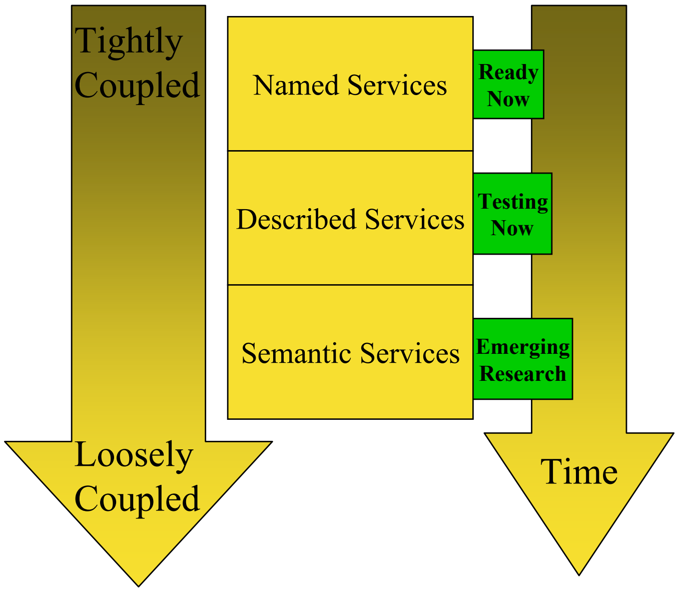
- The TIA Application is the abstraction that links TIA Architecture efforts.
  - **TIA Application** is composed of platform services and is dependent upon use cases.
  - **Process** maps into functional and user requirements.
  - **Service** maps into Grid Services and components developed within IAO and others (e.g., CIM, SEAS)



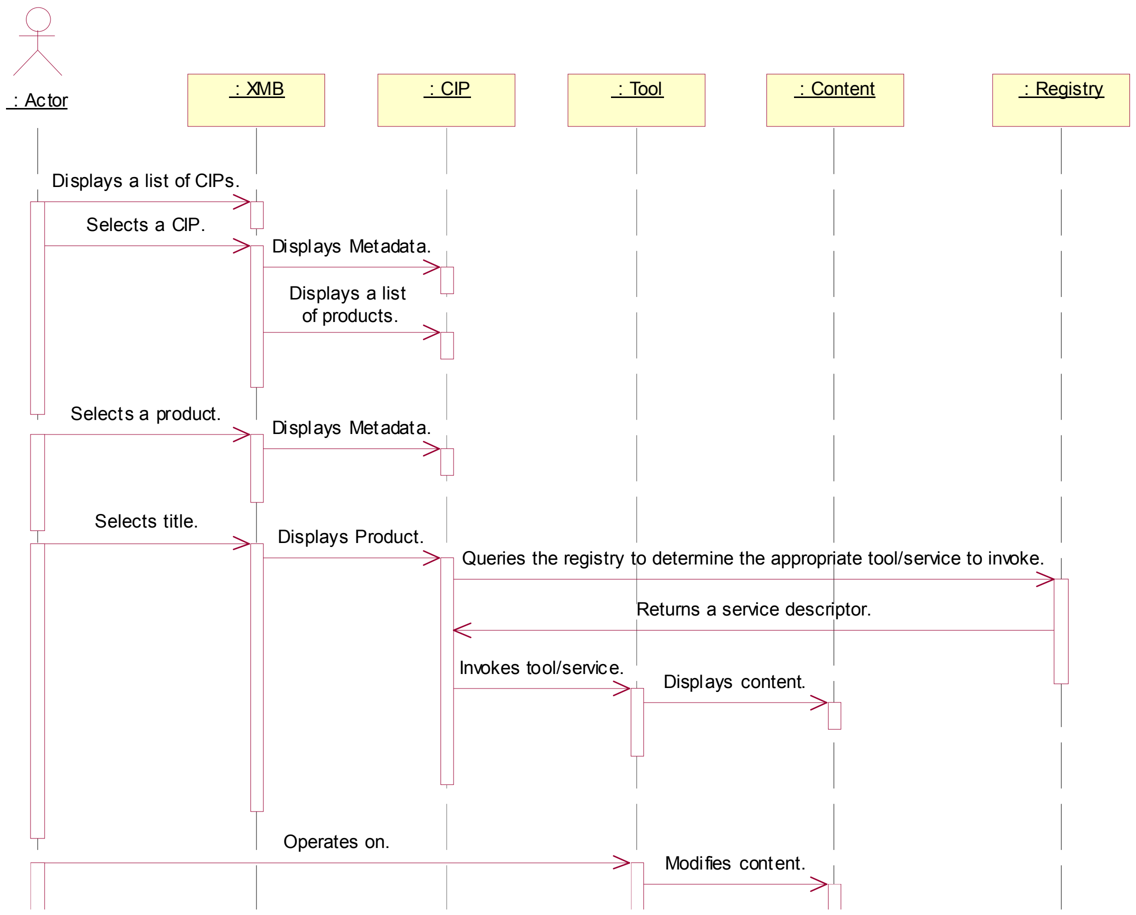
TIA Applications are inherently services-based. ...

2.2.3. There are three levels of service coupling.

- Applications provide the UUID of the service they need.
  - *Named Services*
- Applications provide the syntactic description of the service they need (e.g., QoS).
  - *Described Services*
- Applications provide the semantics to discover the service they need.
  - *Semantic Services*



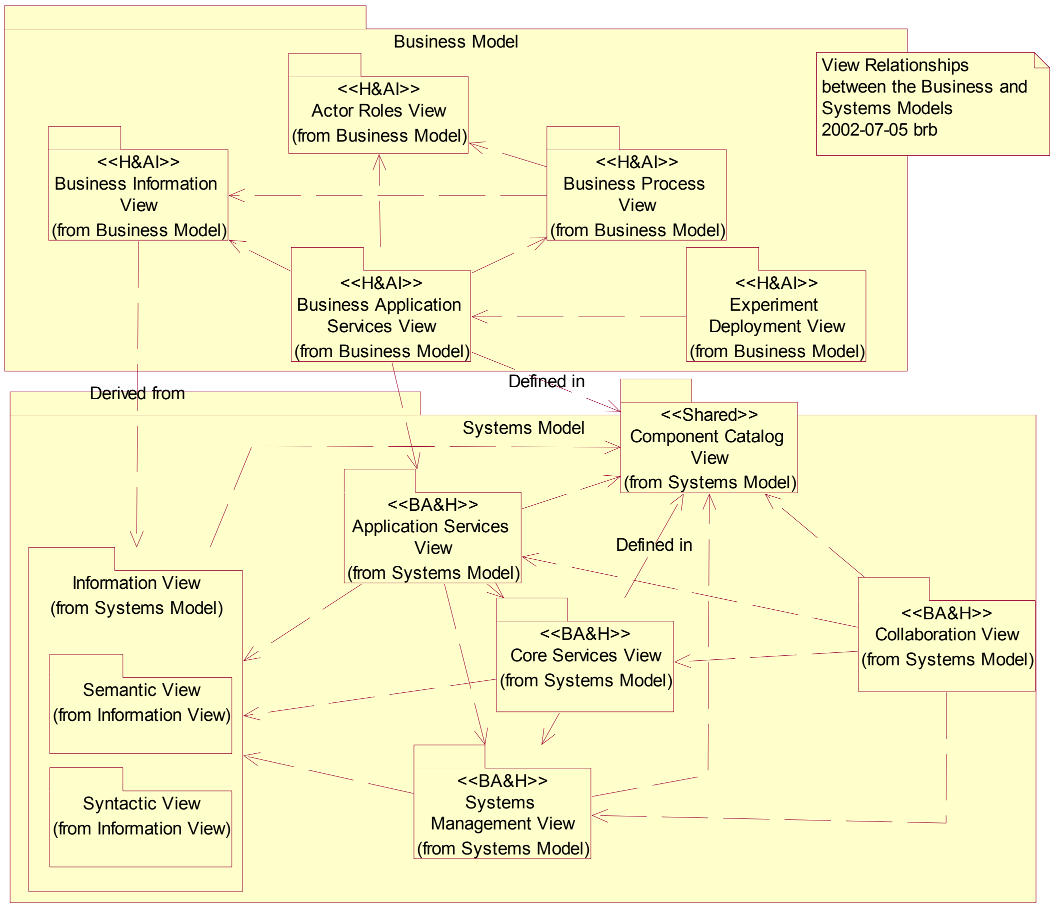
2.2.3.1 Service Application Example



An example of a "Service Application" using the XMB

3. TIA System Description Overview

The TIA System Description contains two models, a Business Model and Systems Model. Each model is partitioned into views as shown below. The linking between the models is shown in the Business Information View and the Business Application Services View. The responsibility for the implementation of the TIA System Description is partitioned by dividing the views within the models. A table of development responsibilities is presented in the table below.


Business Model
View Relationships between the Business and Systems Models 2002-07-05 brb
<<H&AI>> Actor Roles View (from Business Model)
<<H&AI>> Business Information View (from Business Model)
<<H&AI>> Business Process View (from Business Model)
<<H&AI>> Business Application Services View (from Business Model)
<<H&AI>> Experiment Deployment View (from Business Model)
Defined in
Derived from
<<Shared>> Component Catalog View (from Systems Model)
Systems Model
<<BA&H>> Application Services View (from Systems Model)
Defined in
Information View (from Systems Model)
Semantic View (from Information View)
Syntactic View (from Information View)
<<BA&H>> Core Services View (from Systems Model)
<<BA&H>> Collaboration View (from Systems Model)
<<BA&H>> Systems Management View (from Systems Model)

TIA System Description Models

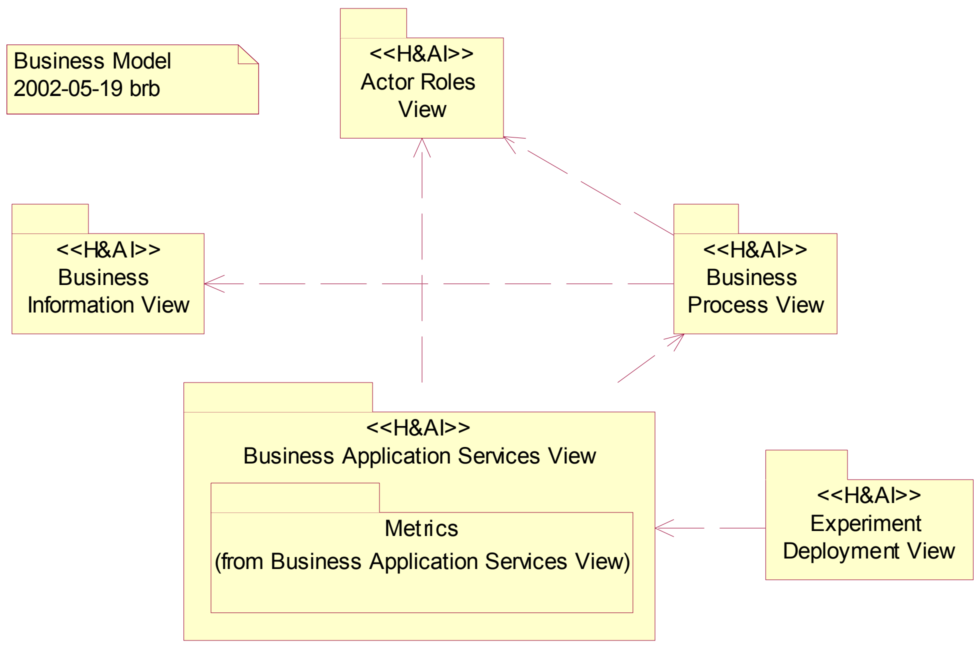
Hicks and Associates is responsible for:
- Actors Roles View
- Business Process View
- Business Information View
- Business Application Services View
- Experiment Deployment View

Booz-Allen & Hamilton, Inc. is responsible for:
- Application Services View
- Core Services View
- Systems Management View
- Collaboration View

Shared Packages
- Information View
- Component Catalog View

3.1. Business Model

The Business Model is a description of the tasks and activities, operational elements, metrics, and information flows required to accomplish or support operations. It contains representations of the operational elements, assigned tasks and activities, metrics, and information flows. It relates the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges to the operational use cases. The Business Model contains five views: (1) Actor Roles View, (2) Business Process View, (3) Business Information View, (4) Business Application Service View, and (5) Experiment Deployment View.

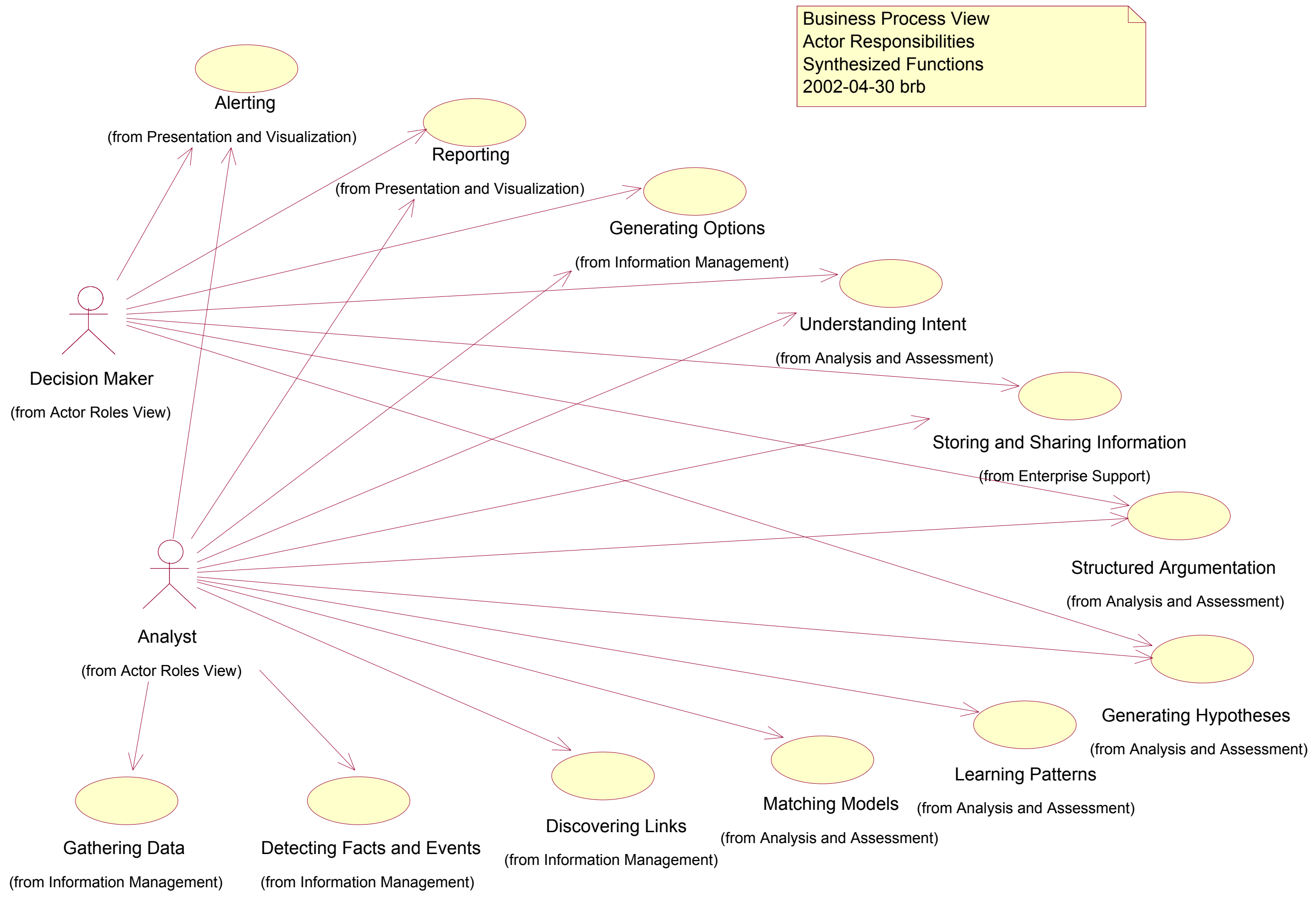

Business Model
2002-05-19 brb
<<H&AI>>
Actor Roles View
<<H&AI>>
Business Information View
<<H&AI>>
Business Process View
<<H&AI>>
Business Application Services View
Metrics
(from Business Application Services View)
<<H&AI>>
Experiment Deployment View

Business Model Views

3.1.1. Actor Roles View

The Actors Roles View contains the identification and description of roles within the TIA System. A role, in this sense, consists of a set of responsibilities where responsibilities are a collection of goals, constraints, and rules. The aggregation of the responsibilities contained in the Actor Roles View is intended to be the complete set required to accomplish the goals of the implementing organization.

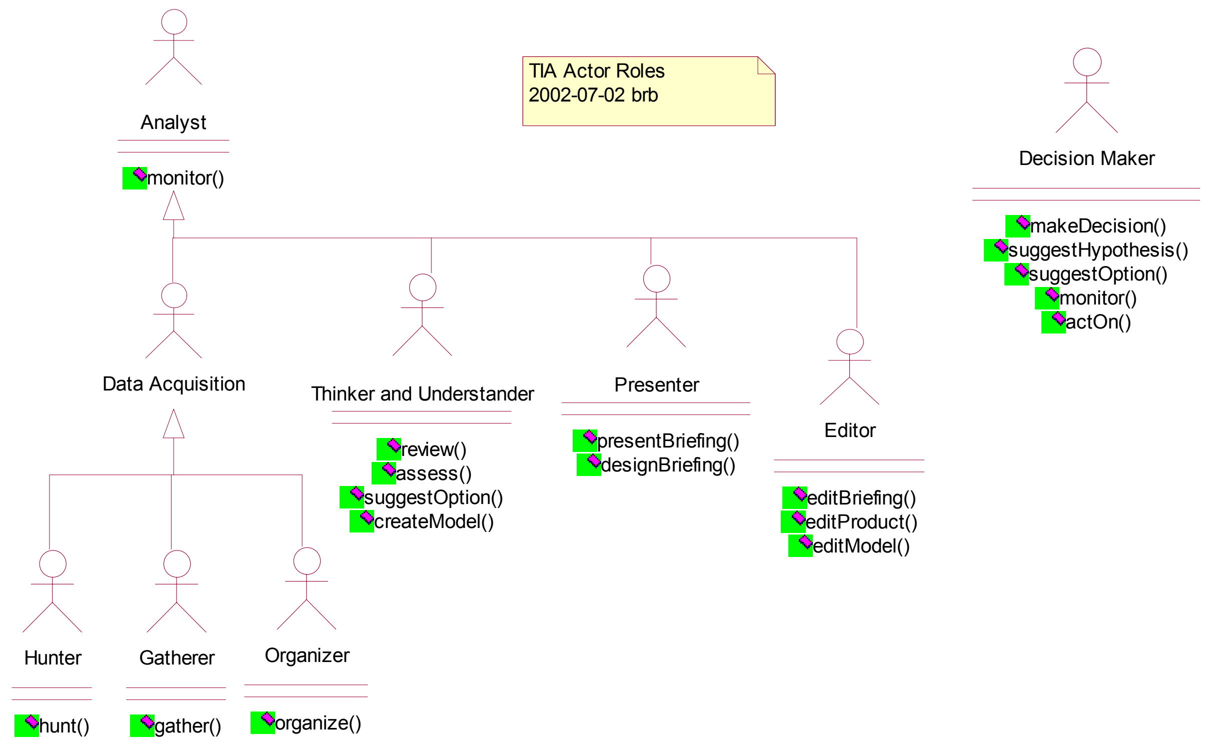
3.1.1.1 Actor (Role) Functional Responsibilities


Business Process View
Actor Responsibilities
Synthesized Functions
2002-04-30 brb
Alerting
(from Presentation and Visualization)
Reporting
(from Presentation and Visualization)
Generating Options
(from Information Management)
Understanding Intent
(from Analysis and Assessment)
Storing and Sharing Information
(from Enterprise Support)
Structured Argumentation
(from Analysis and Assessment)
Generating Hypotheses
(from Analysis and Assessment)
Learning Patterns
(from Analysis and Assessment)
Matching Models
(from Analysis and Assessment)
Discovering Links
(from Information Management)
Detecting Facts and Events
(from Information Management)
Gathering Data
(from Information Management)
Decision Maker
(from Actor Roles View)
Analyst
(from Actor Roles View)

TIA System Role Functional Responsibilities

3.1.1.2 TIA Roles

An actor within the system may assume multiple roles. For example, a Decision Maker can assume the role of a Presenter and Editor. The combination of responsibilities represented by each role is modeled as operations on the Actor Roles as shown below.



TIA System Functional Roles

3.1.1.2.1 Analyst Role

The Analyst Role monitors a goal and is the parent role for all other analyst subroles within the system.

3.1.1.2.1.1 Data Acquisition Specialist

The Data Acquisition Specialist Role is specialized into three roles:

1. Hunter
2. Gatherer
3. Organizer

3.1.1.2.1.2 Thinker and Understander

The Thinker and Understander Role reviews analyses, assesses their importance, suggests options, and creates models.

3.1.1.2.1.3 Presenter

The presenter designs and presents briefings.

3.1.1.2.1.4 Editor

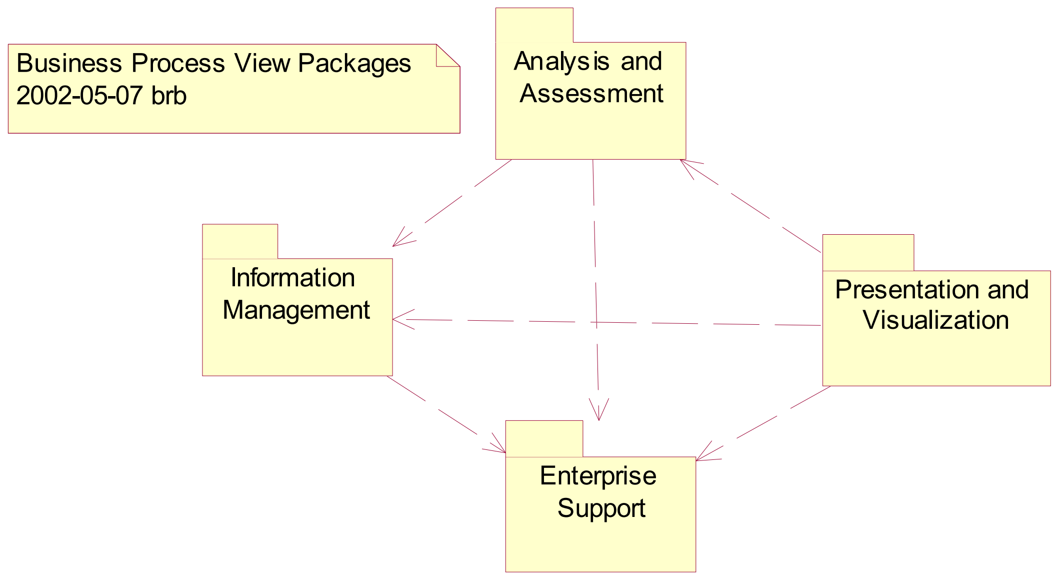
The Editor Role edits briefings, products, and models.

3.1.1.2.2 Decision Maker Role

The Decision Maker Role makes decisions, suggests hypotheses, suggests options, monitors a goal, and acts on the outputs of the analysis process.

3.1.2. Business Process View

The Business Process View provides the reference model for the functional operations. It defines a set of use cases representing the TIA Functional model and maps processes and workflow into those use cases. It is built from use cases and models the end-to-end functionality of the system. It can contain Activity Diagrams or Sequence Diagrams associated with individual Use Cases. Metrics for evaluating technology, as they are developed, will be associated with the Use Cases.



Business Process View Packages

The Business Process View contains four packages representing high-level groups of operations: (1) Analysis, (2) Presentation and Visualization, (3) Enterprise Support, and (4) Information Management. Part of the Business Process View (BPV) captures the actual and planned functional processes that the TIA System supports. This section presents these initial high level uses derived and captured in the BPV. This section also provides a "coverage map" to link existing technology components to the canonical use cases developed as part of the TIA System Architecture Business Process View. The coverage map is provided in two forms: (1) Canonical Use Case Diagrams and (2)

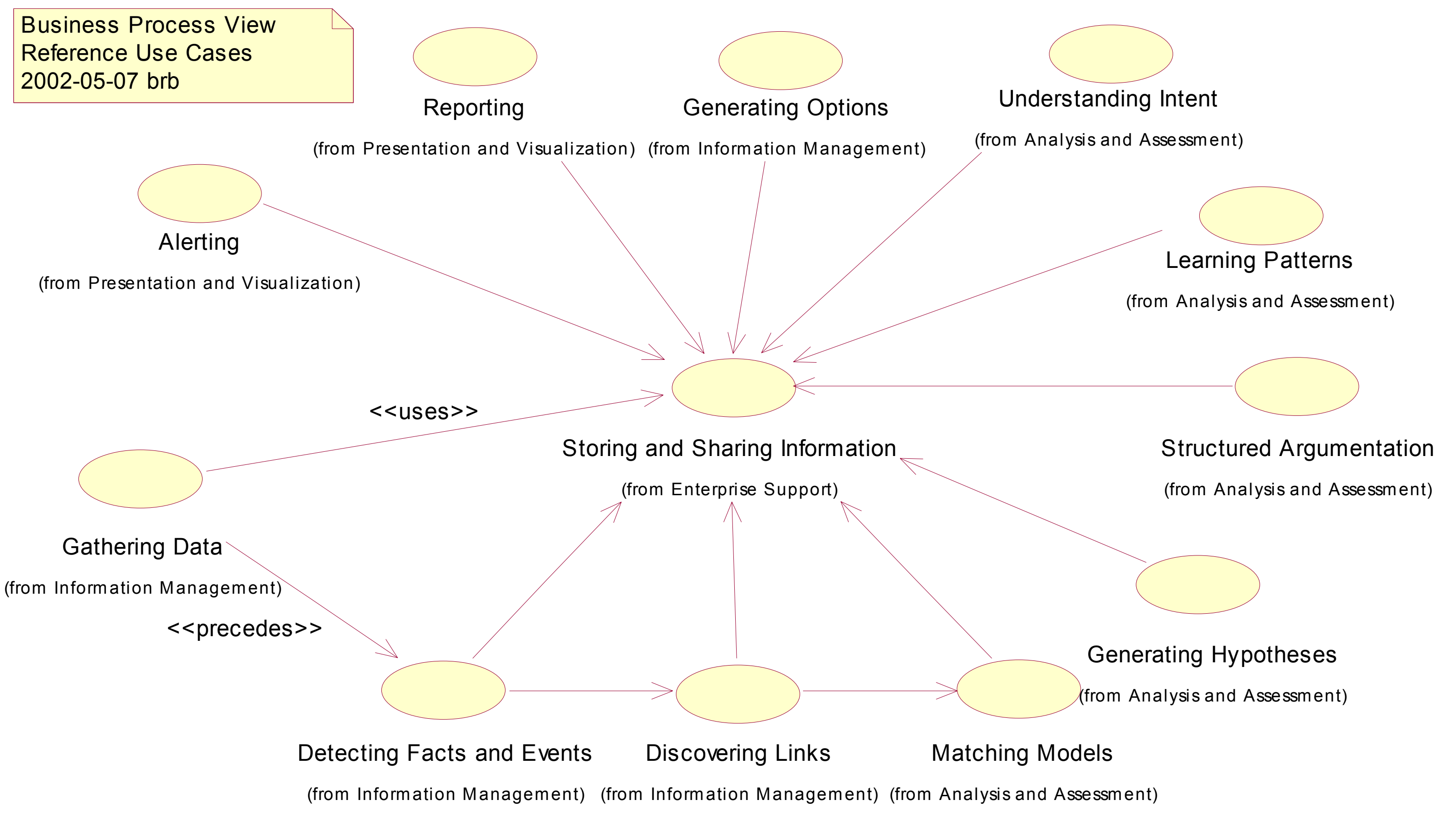
Coverage Map Table. Both versions utilize the color-coding shown below. For the purposes of this document, the technology components considered have been scoped to examples from past or current DARPA Programs, COTS Technologies currently under consideration by DARPA, and contractor-developed technology current in use at INSCOM (IAC). It is intended as a partially representative, rather than exhaustive, map of technologies to use cases. As a result, some use cases may be associated with more than one technical component and others may have a single component (or none) that only partially cover the required functionality.

TBA	TBA area has been identified by the TIA Team as a class of technology components associated with the functional Use Case that have yet to be acquired or developed.
Developing	Technology Component is in the **Developing State**. It has been identified by the TIA Team and its development progress is being monitored.
Prototyping	Technology Component is in the **Prototyping State**. It has been acquired by the TIA Team and is actively being applied to the TIA System. *Metrics* are under development.
Experimenting	Technology Component is in the **Experimenting State**. It has been selected for experimentation by the TIA Team. *Measurements* are being made according to *metrics* where indicated in the class attributes.
Transitioning	Technology Component is **Transitioning**. Technology has been validated by metrics and users and is being transitioned into operational use. *Measured* values for *metrics* are being verified.

Technology Readiness Color Codes

3.1.2.1 Reference Use Cases

The TIA Reference Use Cases, shown below, represent the Use Cases and analytical process supported by the TIA System. They contain aspects synthesized from multiple processing models (e.g., the ASW Signal Processing Model). As a Use Case model, it represents the relationships between Use Cases as opposed to a sequential flow. Business Processes exist across configurations of the Use Cases.

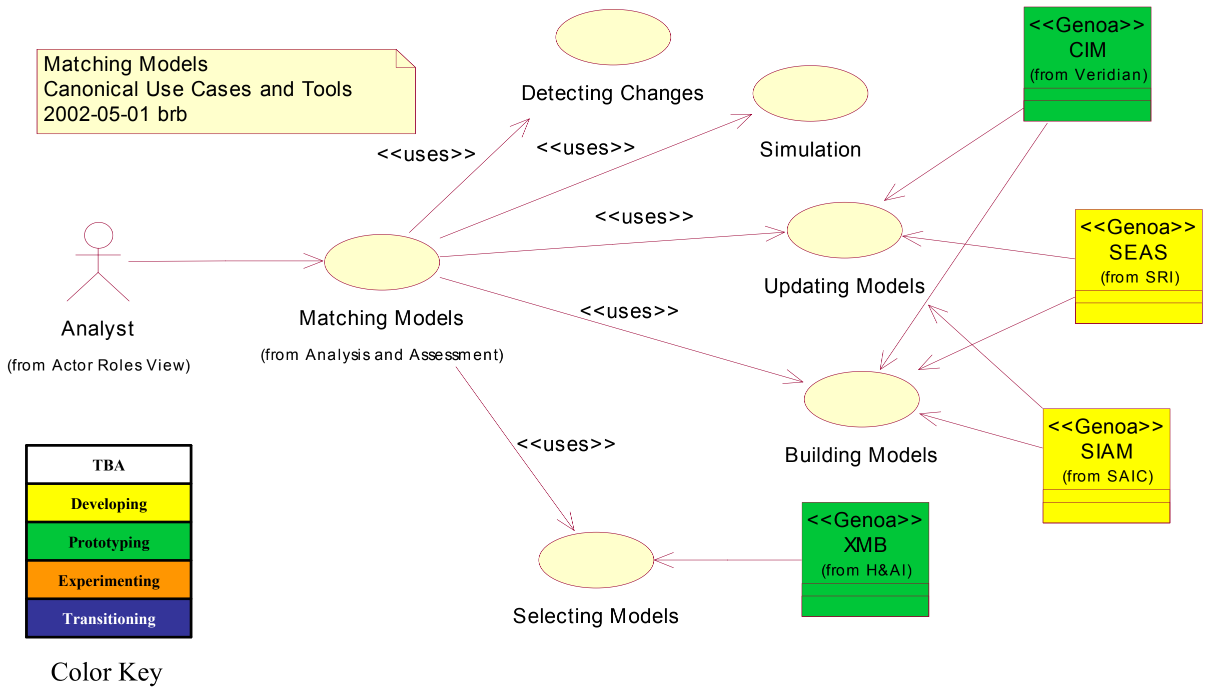

Business Process View
Reference Use Cases
2002-05-07 brb
Reporting
(from Presentation and Visualization)
Generating Options
(from Information Management)
Understanding Intent
(from Analysis and Assessment)
Alerting
(from Presentation and Visualization)
Learning Patterns
(from Analysis and Assessment)
<<uses>>
Storing and Sharing Information
(from Enterprise Support)
Structured Argumentation
(from Analysis and Assessment)
Gathering Data
(from Information Management)
<<precedes>>
Generating Hypotheses
(from Analysis and Assessment)
Detecting Facts and Events
(from Information Management)
Discovering Links
(from Information Management)
Matching Models
(from Analysis and Assessment)

TIA System Functional Processing Flow

3.1.2.2 Analysis and Assessment

3.1.2.2.1 Matching Models

Matching Models represents the capability for analysts to select, build, update, simulate, and detect changes within models representing activities of interest.



Matching Models Canonical Use Cases

3.1.2.2.1.1 Detecting Changes

Detecting Changes represents the capability for analysts to monitor fully or partially populated model instantiations based on new information entering the system, implicitly or explicitly, to determine when and how state changes within the models occur.

3.1.2.2.1.2 Simulation

Simulation represents the capability for analysts to create simulations of the effects of new information and links on past, present, or future scenarios.

3.1.2.2.1.3 Updating Models

Updating Models represents the capability for analysts to insert new information into both models and model instantiations.

3.1.2.2.1.4 Selecting Models

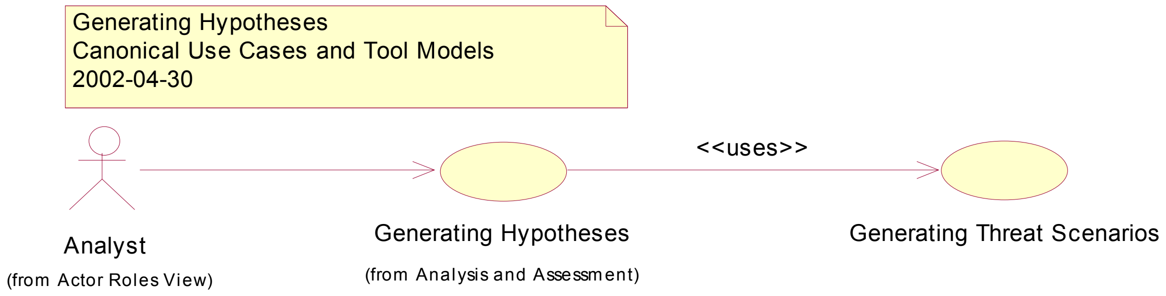
Selecting Models represent the capability for analysts to determine a model (or set of models) that is potentially useful for understanding a specific situation. It includes both manual and automatic selection.

3.1.2.2.1.5 Building Models

Building Models represents the capability for analysts to create models, capturing expert knowledge about methods to achieve a strategic goal (e.g., CW/BW Acquisition Models) and model instantiations, which is the population of a model with specific information.

3.1.2.2.2 Generating Hypotheses

Generating Hypotheses represents the capability for analysts to create scenarios and potential explanations of a complete or partial set of facts, events, links, and models.


Generating Hypotheses
Canonical Use Cases and Tool Models
2002-04-30
<<uses>>
Analyst
(from Actor Roles View)
Generating Hypotheses
(from Analysis and Assessment)
Generating Threat Scenarios

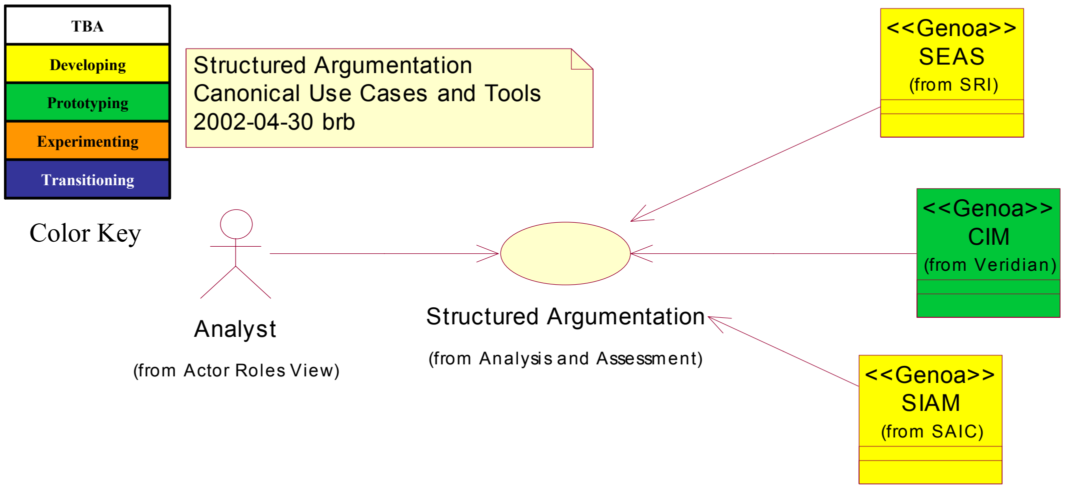
Generating Hypotheses Canonical Use Cases

3.1.2.2.2.1 Generating Threat Scenarios

Generating Threat Scenarios represents the capability for analysts to create scenarios representing possible threats that have potential to occur given a partial set of facts, events, links, and models.

3.1.2.2.3 Structured Argumentation

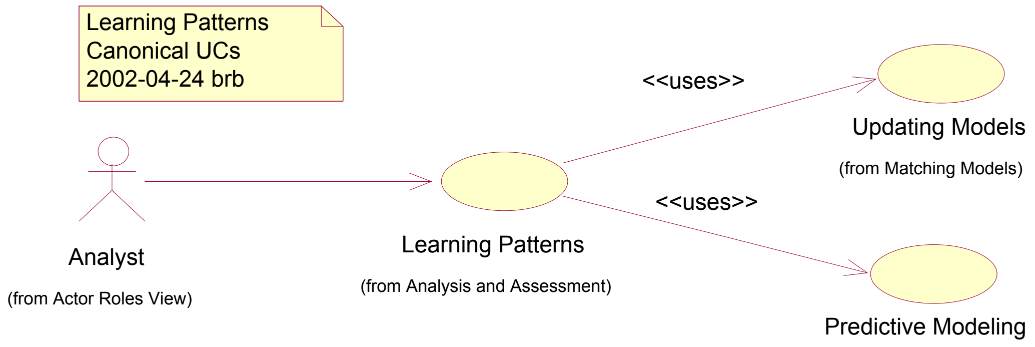
Structured Argumentation represents the capability for analysts to use models, which capture expert knowledge, to reason about potential threats, intent, and the like.



Structured Argumentation Canonical Use Cases

3.1.2.2.4 Learning Patterns

Learning Patterns represents the capability for analysts to associate sequences of facts, events, links, and models with known activities. It includes the leveraging of prior knowledge to explain current situations.



Learning Patterns Canonical Use Cases

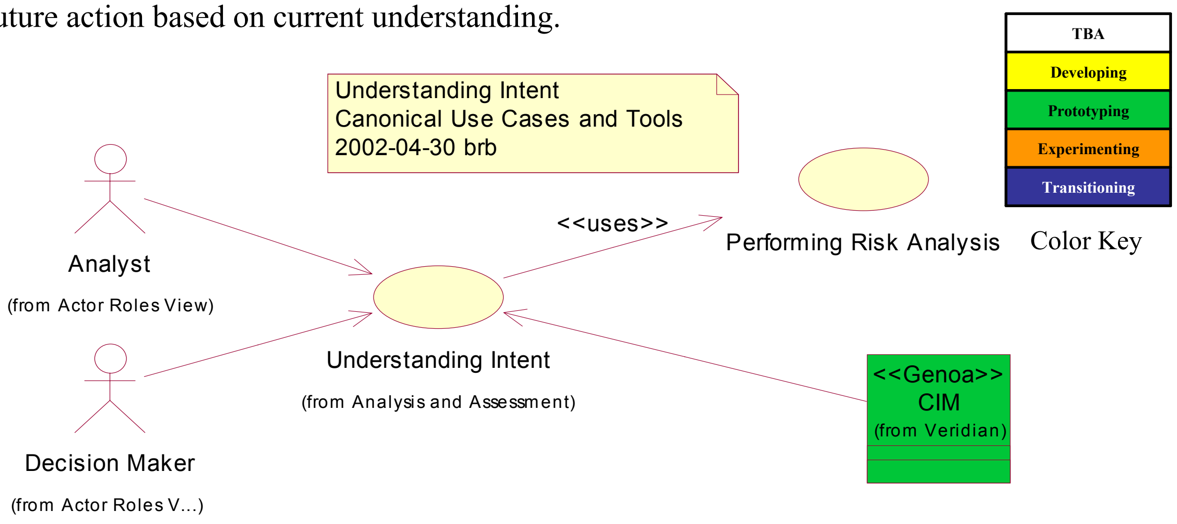
3.1.2.2.4.1 Updating Models

3.1.2.2.4.2 Predictive Modeling

Predictive Modeling represents the capability for analysts to create and instantiate models representing the future effects based on current understanding.

3.1.2.2.5 Understanding Intent

Understanding Intent represents the capability for analysts and decision-makers to develop an understanding of the intent of an individual or set of individuals with regards to a future action based on current understanding.



Understanding Intent Canonical Use Cases

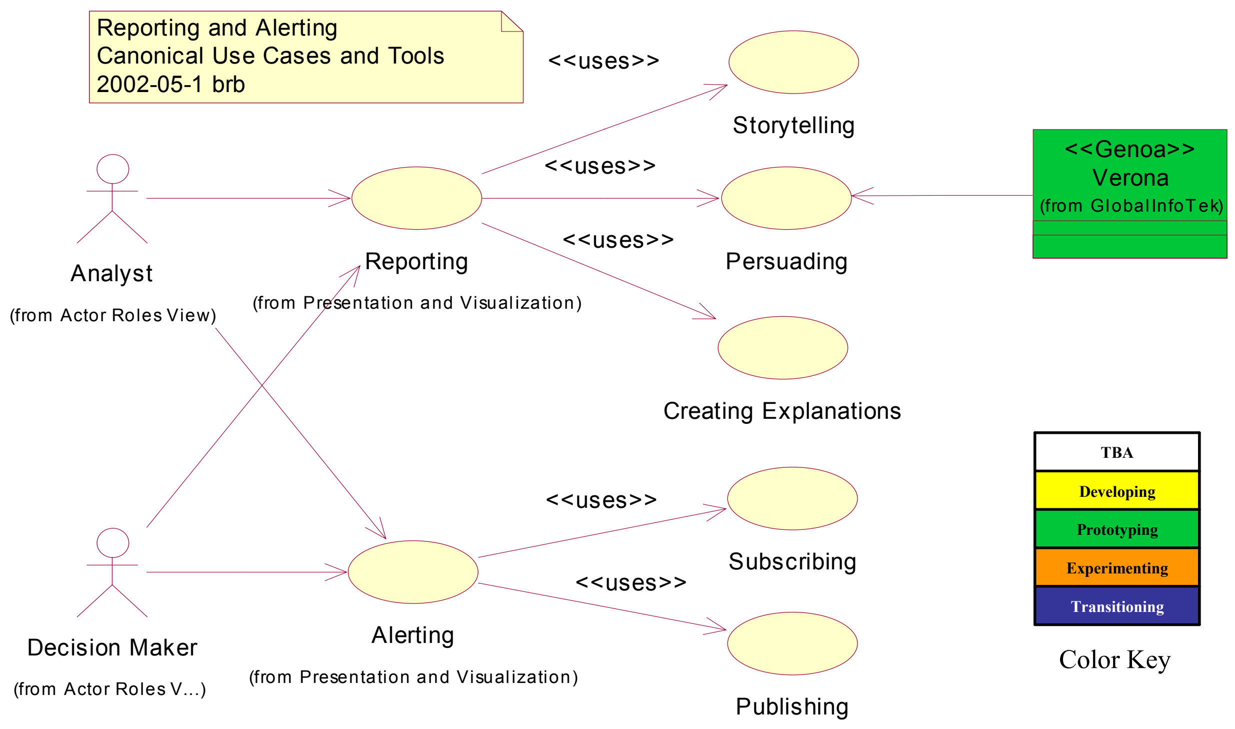
3.1.2.2.5.1 Performing Risk Analysis

Performing Risk Analysis represents the capability for analysts and decision-makers to evaluate the current understanding of intent with respect to the risk the achievement (or lack) of that intent creates. This analysis crosses multiple domains including military, civilian, policy, and so on.

3.1.2.3 Presentation and Visualization

3.1.2.3.1 Reporting and Alerting

Reporting and Alerting represents the capability for analysts and decision-makers to communicate and understand the results of analysis performed. It represents the process of managing the outputs of analysis. In addition, it represents the capabilities for analysts and decision-makers to receive "alerts" when the outputs of the analysis process are of critical interest.



Reporting and Alerting Canonical Use Cases

3.1.2.3.1.1 Storytelling

Storytelling represents the capabilities for analysts and decision-makers to use storytelling and narrative techniques to communicate analysis output. Conveying information in a story provides a rich context, remaining in the conscious memory longer and creating more memory traces than decontextualized information. Thus, a story is more likely to be acted upon than "normal" means of communication. Storytelling, whether in a personal or organizational setting, connects people, develops creativity, and increases confidence. The use of stories in organizations can build descriptive capabilities, increase organizational learning, convey complex meaning, and communicate common values and rule sets.

3.1.2.3.1.2 Persuading

Persuading represents the capabilities for analysts and decision-makers to present the analysis output and convince the intended audience of the validity, options, and the like, of the results.

3.1.2.3.1.3 Creating Explanations

Creating Explanations represents the capabilities for analysts and decision-makers to aggregate the set of information used in analysis into a coherent explanation of intent, a particular event, and the like.

3.1.2.3.1.4 Subscribing

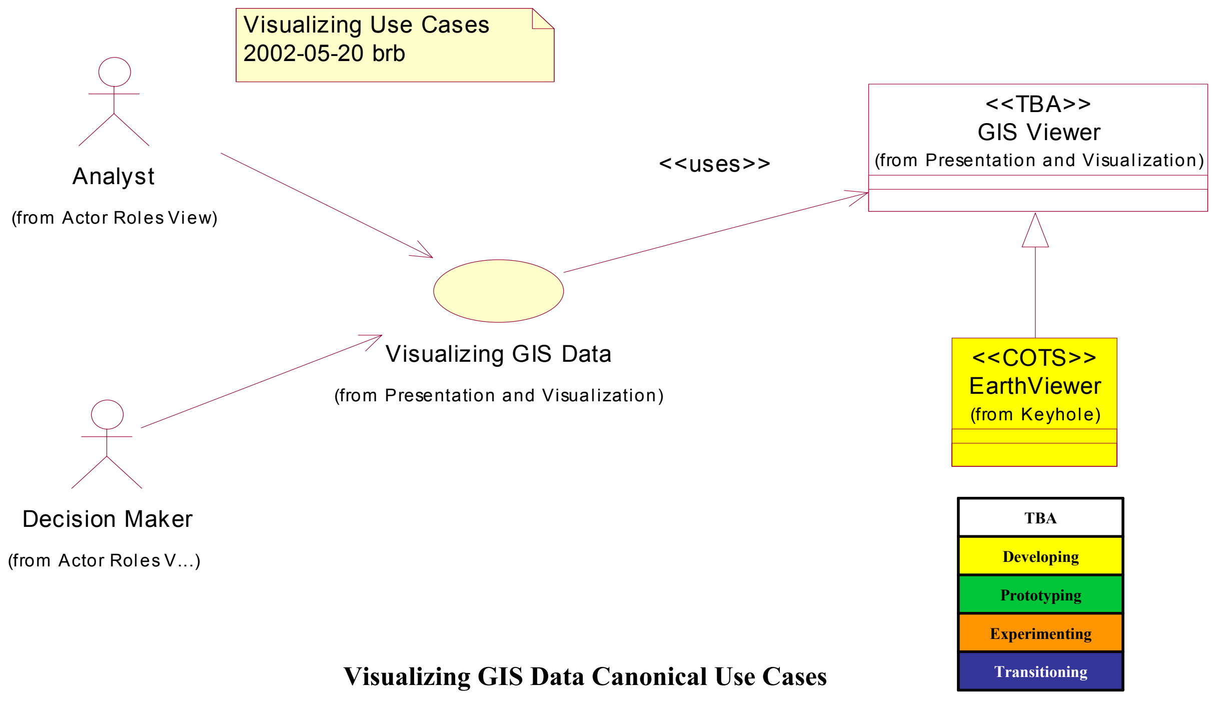
Subscribing represents the capabilities for an analyst and decision-makers to register to receive alerts when outputs of the intelligence process correspond to a particular area of interest.

3.1.2.3.1.5 Publishing

Publishing represents the capabilities for delivering alerts to analysts and decision-makers who have subscribed to a particular area of interest.

3.1.2.3.2 Visualizing GIS Data

Visualizing GIS Data represents the capability for analysts and decision-makers to view and navigate GIS Information.

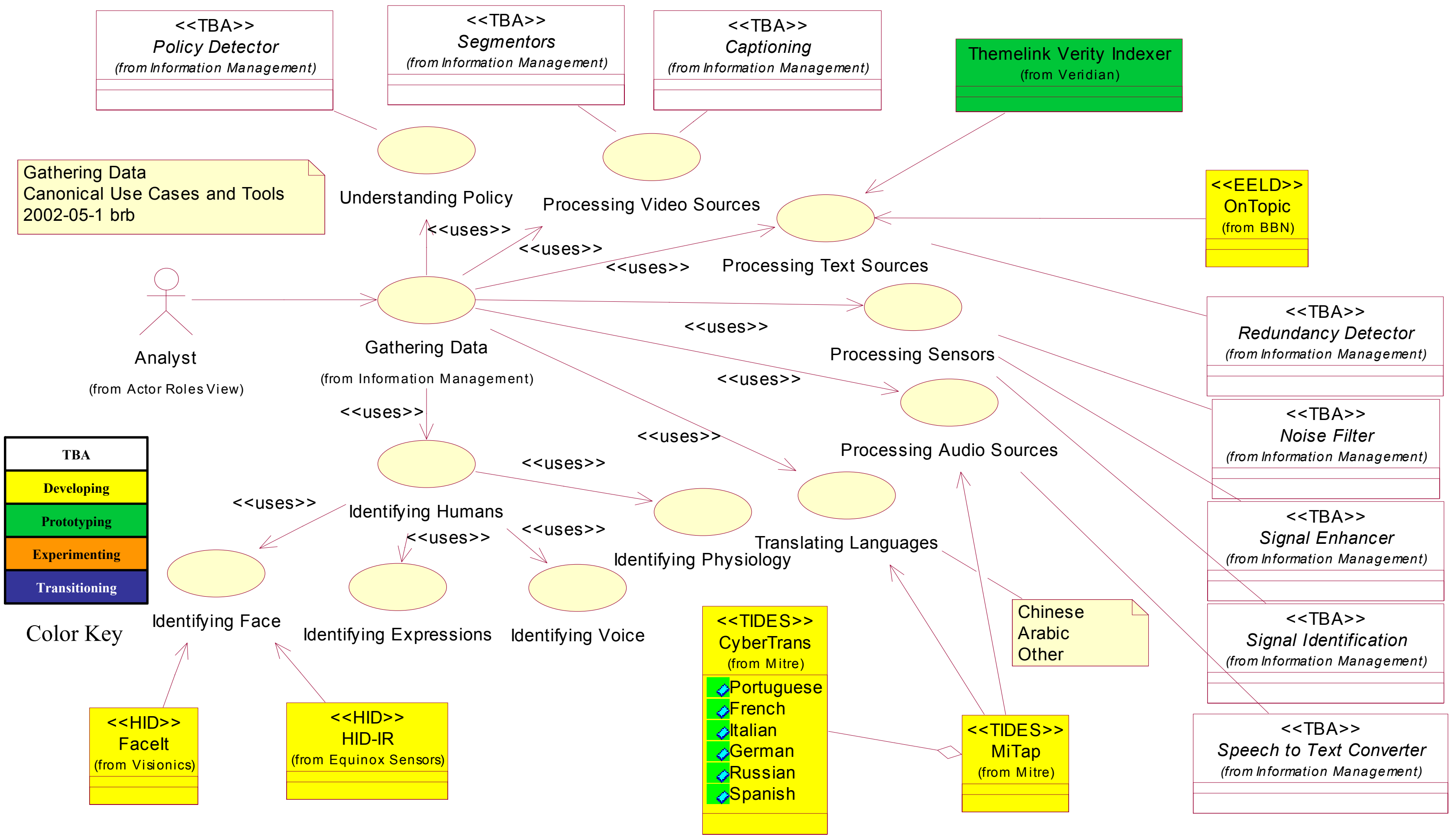


Visualizing GIS Data Canonical Use Cases

Color Key

3.1.2.4 Information Management

3.1.2.4.1 Gathering Data



Gathering Data Canonical Use Cases

3.1.2.4.1.1 Understanding Policy

Understanding Policy represents the capability for analysts performing Gathering Data activities to understand and adhere to the current policies regarding data gathering, monitoring sources and individuals, etc. Understanding Policy is included in all aspects of Gathering Data. There is one class of tool specifically identified as capabilities for Understanding Policy:

a. Policy Detector

3.1.2.4.1.2 Processing Sources

Processing Sources represents the capability for analysts to review and process the large numbers of information sources entering the system. There are three Use Cases of processing sources specifically identified:

1. Processing Video Sources: There are two classes of technology components specifically identified as capabilities for Processing Video Sources:
   a. Segmentors
   b. Captioning
2. Processing Text Sources: There are two classes of technology components specifically identified as capabilities for Processing Text Sources:
   a. Indexers
   b. Redundancy Detector
3. Processing Sensors: There are three classes of technology components specifically identified as capabilities for Processing Sensors:
   a. Noise Filter
   b. Signal Enhancer
   c. Signal Identification
4. Processing Audio Sources: There is one class of tool specifically identified as capabilities for Processing Audio Sources:
   a. Speech to Text Converter

3.1.2.4.1.3 Translating Languages

Translating Languages represents the capability for analysts to find and interpret needed information, quickly and effectively, regardless of language or medium. A huge volume and variety of speech and text is available electronically in many forms (e.g., news broadcasts, newswire, World Wide Web, Internet). This raw data flows and grows constantly, spanning many languages. A great deal of vital information exists only in, or appears sooner in, languages other than English. The Translating Use Case addresses technology that enables English speakers to access this data. Examples of targeted languages include, among others Chinese and Arabic.

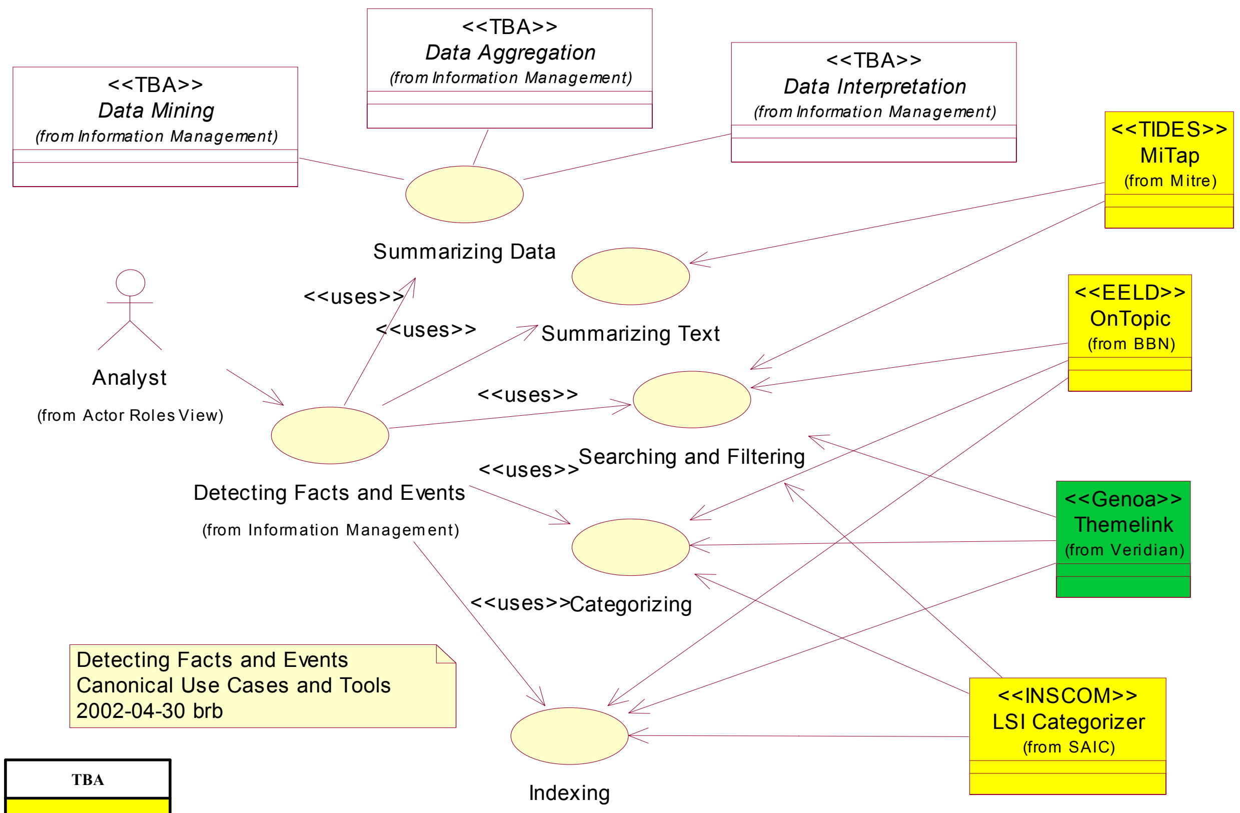
3.1.2.4.1.4 Identifying Humans

Identifying Humans represents the capability for analysts to use multimodal surveillance capabilities for identifying humans in order to enhance the protection of U.S. forces and facilities from terrorist attacks. It enables successfully detecting, classifying, and identifying humans under a variety of conditions including groups, weather, disguises, etc. There are four sub use cases:

1. Identifying Faces
2. Identifying Expressions
3. Identifying Voice
4. Identifying Physiology

3.1.2.4.2 Detecting Facts and Events

Detecting Facts and Events represents the capability for analysts to extract pertinent facts and events collected within the Gathering Data Use Cases. Examples of extracted information include an entity with its attributes, a relationship between two or more entities, or an event with various entities playing roles and/or being in certain relationships.



TBA
Developing
Prototyping
Experimenting
Transitioning

Color Key

Detecting Facts and Event Canonical Use Cases

3.1.2.4.2.1 Summarizing

Summarizing represents the capability for analysts to develop and visualize summary information representing the agreement of facts and events. Summarizing includes visualizations, automated alert systems, and/or automatic document summarizers. There are two use cases of summarizing identified:

1. Summarizing Data: Includes three specific classes of technology components:
   a. Data Mining
   b. Data Aggregation
   c. Data Interpretation
2. Summarizing Text

3.1.2.4.2.2 Searching and Filtering

Searching and Filtering represents the capability for analysts to search and filter information, specifically in this case Facts and Events. Types of searching include free-text searching, Boolean searching, SQL queries, concept-based searching, among others. Filtering represents the capability for analysts to use searching mechanisms to create "views" of collections of Facts and Events. For example, an analyst may wish to see all of the "Events" that occurred in Xinjiang for a certain date range. Searching and Filtering capabilities may be applied to multiple levels of data and information constructs, for example links.

3.1.2.4.2.3 Categorizing

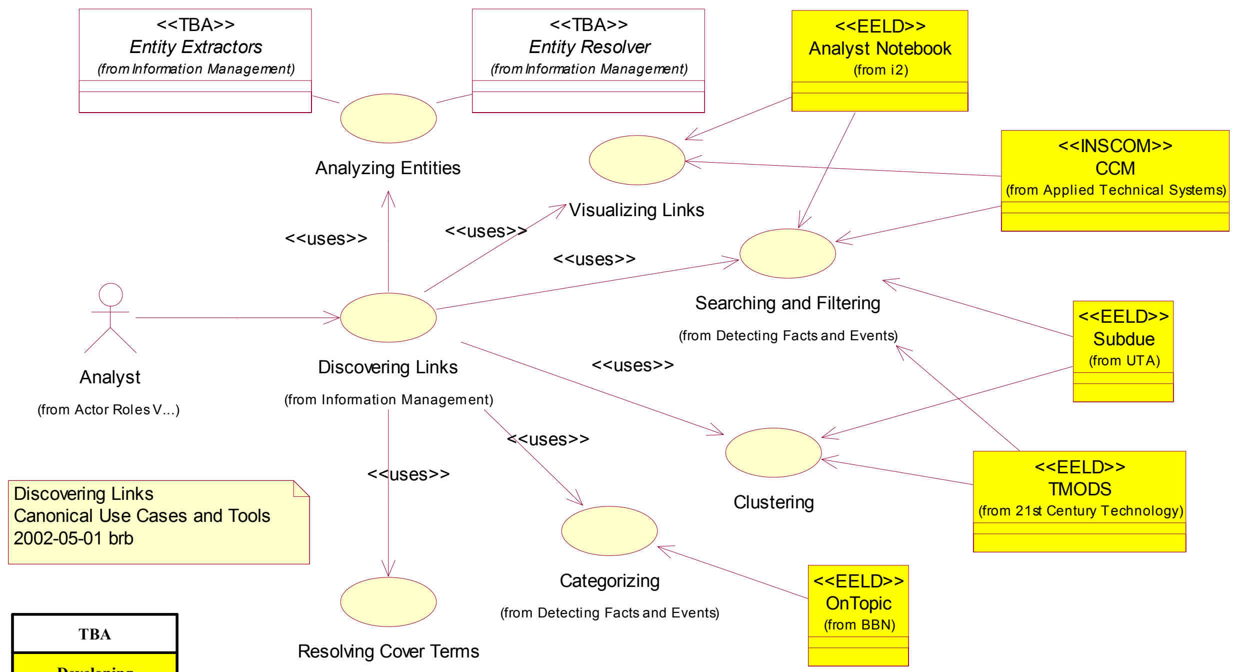
Categorizing represents the capability for analysts to associate sets of facts and/or events into a taxonomy or topic/subject hierarchy. The taxonomy may be defined by the organization or generated automatically as in the case of a Clusterizer.

3.1.2.4.2.4 Indexing

Indexing represents the capability for analysts to store information about facts and events that have been recorded in the system for future retrieval. Indexing may occur at multiple levels including term-indexing and/or concept-indexing.

3.1.2.4.3 Discovering Links

Discovering Links is the ability for an analyst to identify, correlate, and expose relationships between facts and events relevant to a particular area of interest. It creates the ability to find and discover relationships across a breadth of information and between seemingly unrelated items.



Discovering Links Canonical Use Cases

TBA
Developing
Prototyping
Experimenting
Transitioning

Color Key

3.1.2.4.3.1 Analyzing Entities

Analyzing Entities is the ability to search, identify, match, and research entities across aliases and other obfuscating events and tactics. It applies techniques from disciplines such as linguistics, information theory, statistics, probability, cultural anthropology, and formal systems design. There are two classes of technology components specifically identified as capabilities for Analyzing Entities:

a. Entity Extractors

b. Entity Resolver

3.1.2.4.3.2 Visualizing Links

Visualizing Links (or network analysis) is a visualization technique intended to reveal structure in a collection of related records. Link data is modeled using a graph where nodes represent entities and links represent relationships or transactions. It includes aspects of information visualization,[8] the ability to navigate and search "naturally" through unfamiliar information spaces and to manage large-scale and complex data sets.

3.1.2.4.3.3 Searching and Filtering

3.1.2.4.3.4 Clustering

Clustering provides the capability for analysts to group sets of facts or links in a coherent manner to reveal a higher-level link between facts, events, or links.

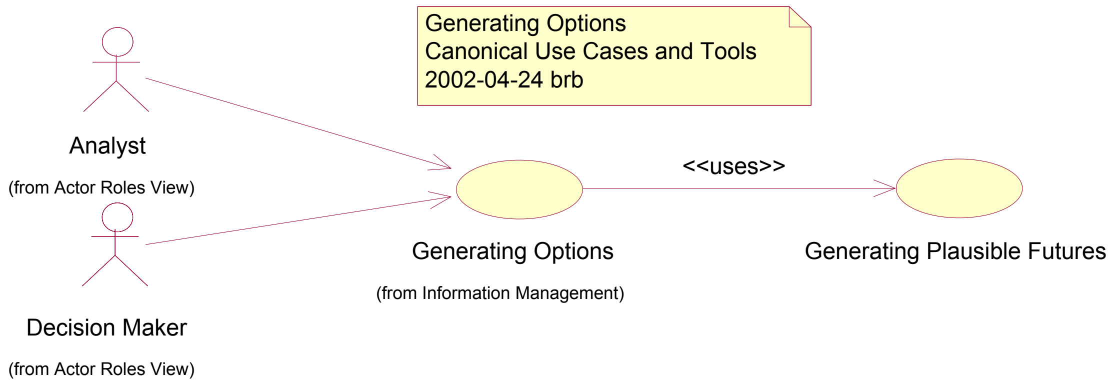
3.1.2.4.3.5 Categorizing

Categorizing provides the capability for analysts to classify sets of facts or links in a user or system-defined hierarchy (e.g., taxonomy) to reveal a higher-level link between facts, events, or links.

3.1.2.4.3.6 Resolving Cover Terms

Resolving Cover Terms provides the capability for analysts to identify coded events and patterns of events across entities. It can be considered a type of pattern-matching for link attributes across time.

3.1.2.4.4 Generating Options



Generating Options Canonical Use Cases

Generating Options represents the capability for analysts and decision-makers to create options for mitigating (and response) based on the current understanding of intent and information gathered.

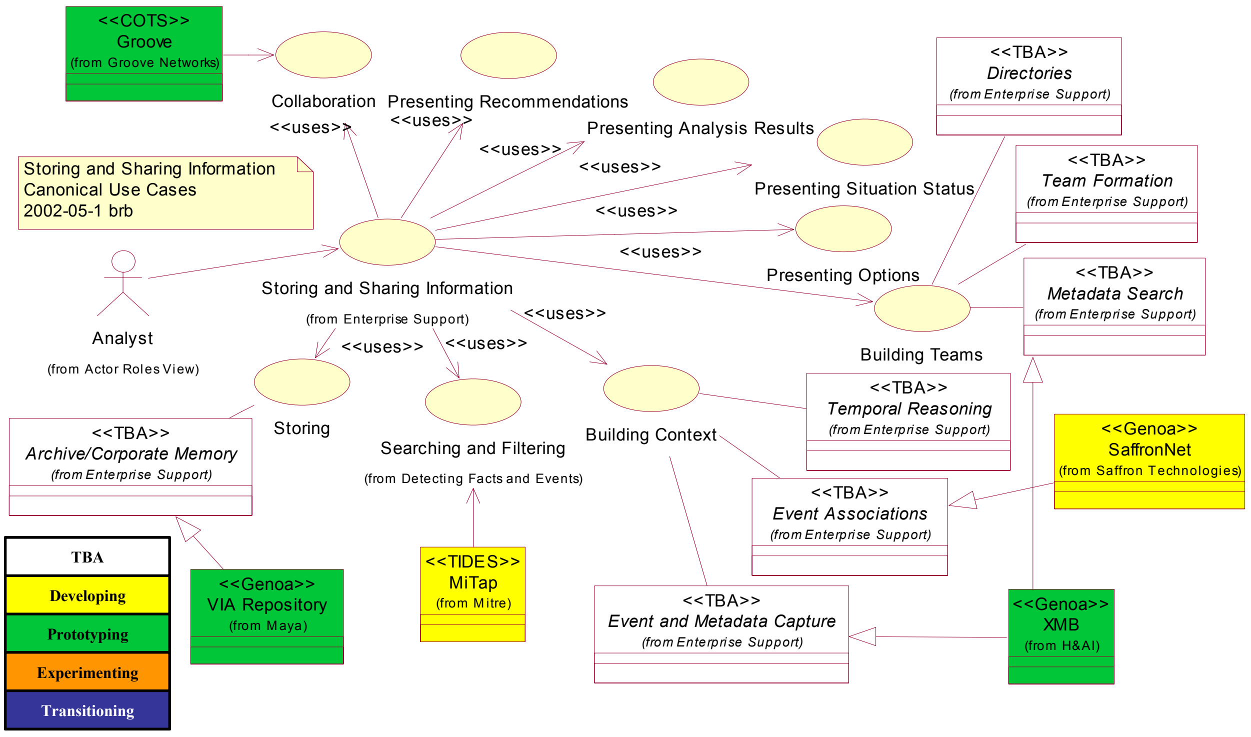
3.1.2.4.4.1 Generating Plausible Futures

Generating Plausible Futures represents the capability for analysts and decision-makers to perform post-event analysis reasoning about the options under consideration. Colloquially, it can be considered the capability to ask, "What will happen if we take this action?"

3.1.2.5 Enterprise Support

3.1.2.5.1 Storing and Sharing Information

Storing and Sharing Information represents the capability for analysts to explicitly and implicitly capture and persist information within the system and exchange information between parties with intersecting operational and functional interests.



Storing and Sharing Information Canonical Use Cases

3.1.2.5.1.1 Collaboration

Collaboration represents the capability for analysts to exchange information and create participatory processes. Collaboration can be implicit, as in sharing information, or explicit, as in a live chat. Furthermore, collaboration can be “polymorphic,” or “deep,” whereby participants use different technology components to operate on the same information structure in a synchronized fashion.

3.1.2.5.1.2 Presenting

Presenting represents the capability for analysts to effectively communicate information to other stakeholders (e.g., decision-makers, other analysts, etc.). It includes visualization, information packaging, information summaries, and other aspects. There are four use cases of presenting identified:

1. Presenting Situation Status

2. Presenting Analysis Results
3. Presenting Options
4. Presenting Recommendations

3.1.2.5.1.3 Building Teams

Building Teams represents the capability for analysts to assemble groups of experts to gain additional input and expertise. Building Teams includes identifying team members and coordinating team activities. There are three classes of technology components specifically identified as capabilities for Building Teams:

a. Directories
b. Team Formation
c. Metadata Search

3.1.2.5.1.4 Building Context

Building Context represents the capability for analysts to capture the context surrounding the information space created by other capabilities. The Core Model for context is shown in the TIA Architecture Fractal Core Model. Context is drawn from a variety of sources and, as a result, has a very high dimensionality of attributes. Furthermore, Context is dynamic. It grows even as analysts handle the content. There are three classes of technology components specifically identified as capabilities for Building Context:

a. Temporal Reasoning
b. Event Associations
c. Event and Metadata Capture

3.1.2.5.1.5 Searching and Filtering

3.1.2.5.1.6 Storing

Storing represents the capability for analysts to persist, retrieve, and operate upon information received from external sources. It includes the ability to store a broad range of information types and provide unified operations and views upon disparate information formats and schemas. There is one classes of technology component specifically identified as capabilities for Storing:

a. Archive/Corporate Memory

3.1.2.6 Canonical Use Case Coverage Map

This section provides a "coverage map" table linking existing technologies to the canonical use cases developed as part of the Business Process View. This section can be considered an alternative view of the technology components and capabilities shown in the preceding section's diagrams. The table uses the color-coding shown below. For the purposes of this document, the technology considered has been scoped to examples from past or current DARPA programs, COTS technologies currently under consideration by DARPA, and contractor-developed technology currently in use at INSCOM (IAC). As this is intended as a partially representative map of technologies to Use Cases, some use cases might be associated with more than one technical component and others might have a single component (or none) that only partially cover the required functionality.

TBA	TBA area has been identified by the TIA Team as a class of technology components that is associated with the functional use case that has yet to be acquired or developed.
Developing	Technology Component is in the **Developing State**. It has been identified by the TIA Team and its development progress is being monitored.
Prototyping	Technology Component is in the **Prototyping** State. It has been acquired by the TIA Team and is actively being applied to the TIA System. *Metrics* are under development.
Experimenting	Technology Component is in the **Experimenting** Stage. It has been selected for experimentation by the TIA Team. **Measurements** are being made according to **Metrics** where indicated in the class attributes.
Transitioning	Technology Component is **Transitioning**. Technology has been validated by metrics and users and is being transitioned into operational use. **Measured** values for *metrics* are being verified.

Technology Readiness Color Code

3.1.2.6.1 *Coverage Map Table*

Functional View	Canonical Use Case	Capability	No.	Program / Tool
Gathering Data	Understanding Policies	Policy Detector	1	
	Preparing Video Sources	Segmentors	2	
		Captioning	3	
	Preparing Text Sources		4	EELD / OnTopic
				Genoa / Themelink Verity Indexer
		Redundancy Detector	5	
	Processing Sensors	Noise Filter	6	
		Signal Enhancement	7	
		Signal Identification	8	
	Processing Audio Sources	Speech to Text Converter	9	TIDES / MiTap
	Translating Languages		10	TIDES / MiTap
				TIDES / CyberTrans Portuguese, French, Italian, German, Russian, Spanish
	Identifying Humans	Identifying Face	11	HID / FaceIt
				HID / HID-IR
		Identifying Expressions	12	
		Identifying Voice	13	
Detecting Facts and Events	Summarizing Data	Data Mining	14	
		Data Aggregation	15	
		Data Interpretation	16	
	Summarizing Text		17	TIDES / MiTap
	Searching and Filtering		18	TIDES / MiTap
				Genoa / ThemeLink
				EELD / OnTopic
				INSCOM / LSI Categorizer
	Categorizing		19	EELD / OnTopic
				INSCOM / LSI Categorizer
				Genoa / ThemeLink

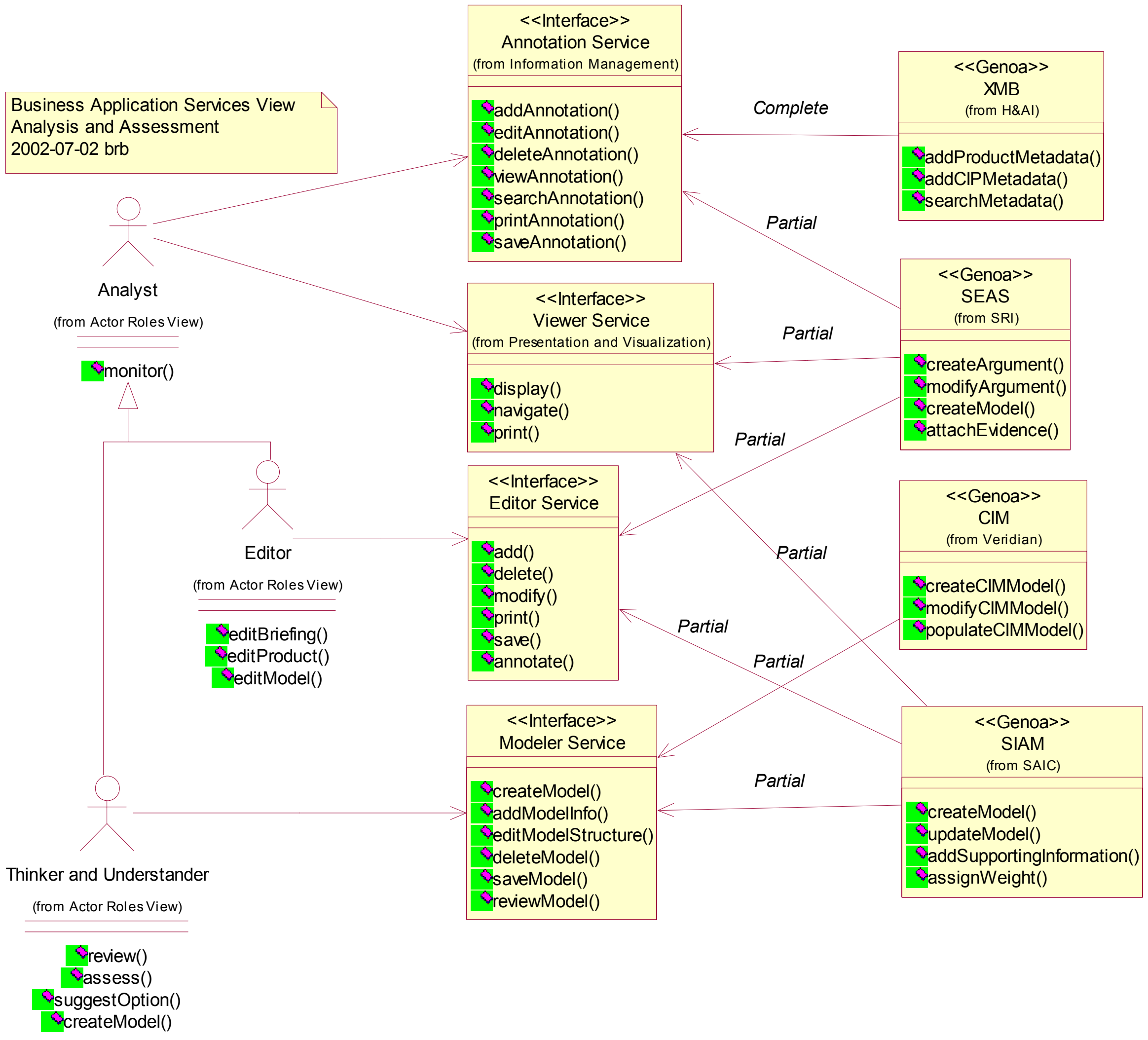
	Indexing		20	EELD / OnTopic Genoa / ThemeLink INSCOM / LSI Categorizer
Storing and Sharing Information	Collaboration		21	Groove
	Presenting Situation Status		22	
	Presenting Analysis Results		23	
	Presenting Options		24	
	Presenting Recommendations		25	
	Building Teams	Directories	26	
		Team Formation	27	
		Metadata Search	28	Genoa / XMB
	Building Context	Temporal Reasoning	29	
		Event Associations	30	Genoa / SaffronNet
		Metadata Capture	31	Genoa / XMB
	Searching and Filtering		32	TIDES / MiTap
	Storing	Archive / Corporate Memory	33	Genoa / VIA Repository
Discovering Links	Analyzing Entities	Entity Extractors	34	
		Entity Resolvers	35	
	Visualizing Links		36	Analyst Notebook (i2) EELD / CCM
	Searching and Filtering		37	EELD / Analyst Notebook (i2) EELD / CCM EELD / Subdue EELD / TMODS
	Clustering		38	EELD / Subdue EELD / TMODS
	Categorizing		39	EELD / OnTopic

	Resolving Cover Terms		40	
Matching Models	Detecting Changes		41	
	Simulation		42	
	Selecting Models		43	XMB
	Building Models		44	Genoa / CIM Genoa / SEAS SIAM
	Updating Models		45	Genoa / CIM Genoa / SEAS SIAM
Generating Hypotheses	Generating Threat Scenarios		46	
Structured Argumentation			47	Genoa / CIM Genoa / SEAS Genoa / SIAM
Learning Patterns	Updating Models		48	
	Predictive Modeling		49	
Understanding Intent	Performing Risk Analysis		50	Genoa / CIM
Generating Options	Generating Plausible Futures		51	
Reporting	Storytelling		52	
	Persuading		53	Genoa / Verona
	Creating Explanations		54	
Alerting	Subscribing		55	
	Publishing		56	
Visualizing GIS Data			57	EarthViewer

3.1.3. Business Application Services View

The Business Application Services View provides the key bridge between the processes in the Business Model and the services in the System Model. It associates Actor Roles (see Section 3.1.1) with Business Application Services, which are modeled as UML Interfaces. The Business Application Services are an aggregate of the operations' specifications (services) and the business rules required for the roles (Actor Roles) to accomplish the activities (use cases). The aggregation of the interfaces associated with a particular use case comprises the capabilities that must be present in the application(s) that implements a use case. A tool or application is thus a realization, complete or partial, of a set of the interfaces.

3.1.3.1 Analysis and Assessment Business Application Services



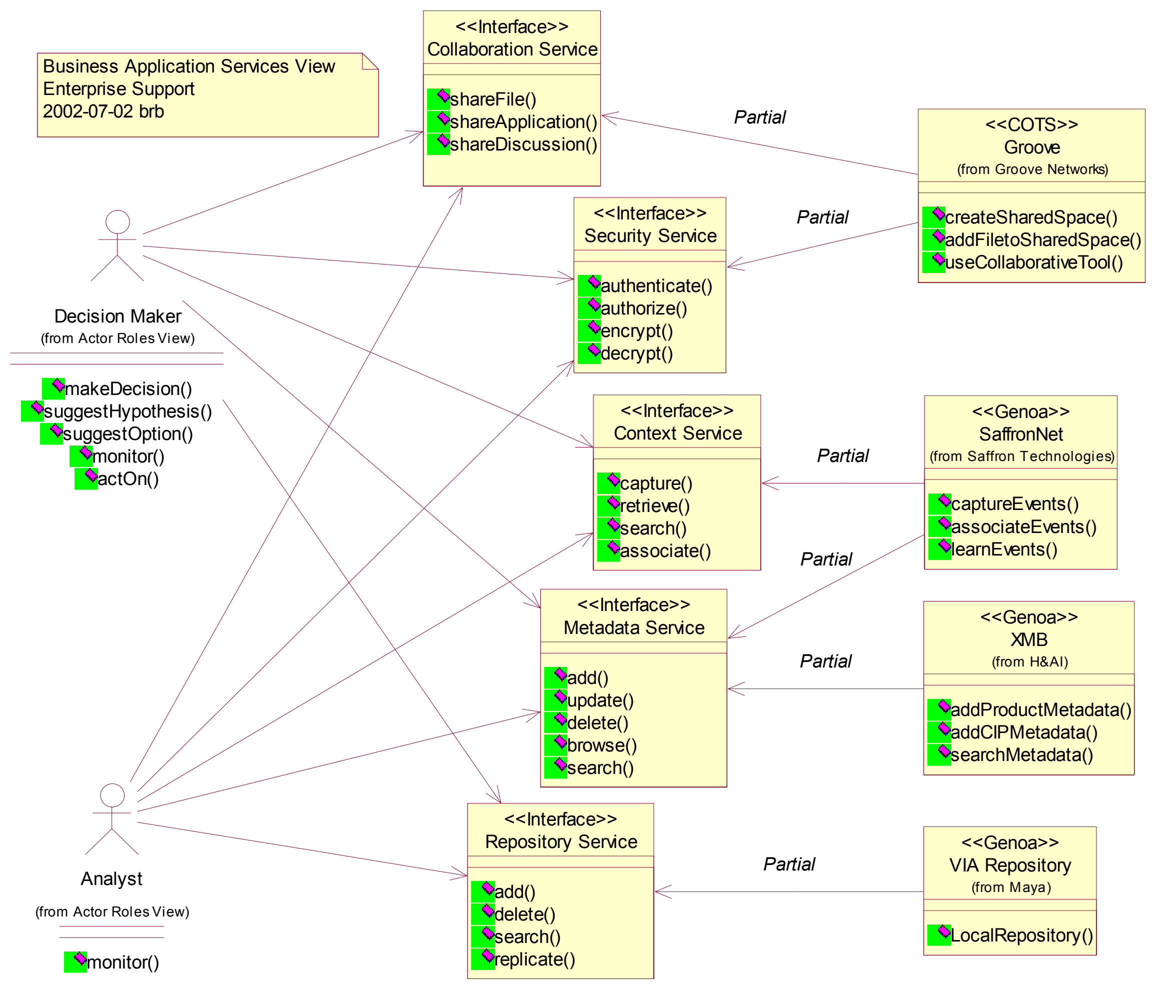
3.1.3.1.1 Annotation Service

3.1.3.1.2 Viewer Service

3.1.3.1.3 Editor Service

3.1.3.1.4 Modeler Service

3.1.3.2 Enterprise Support Business Application Services



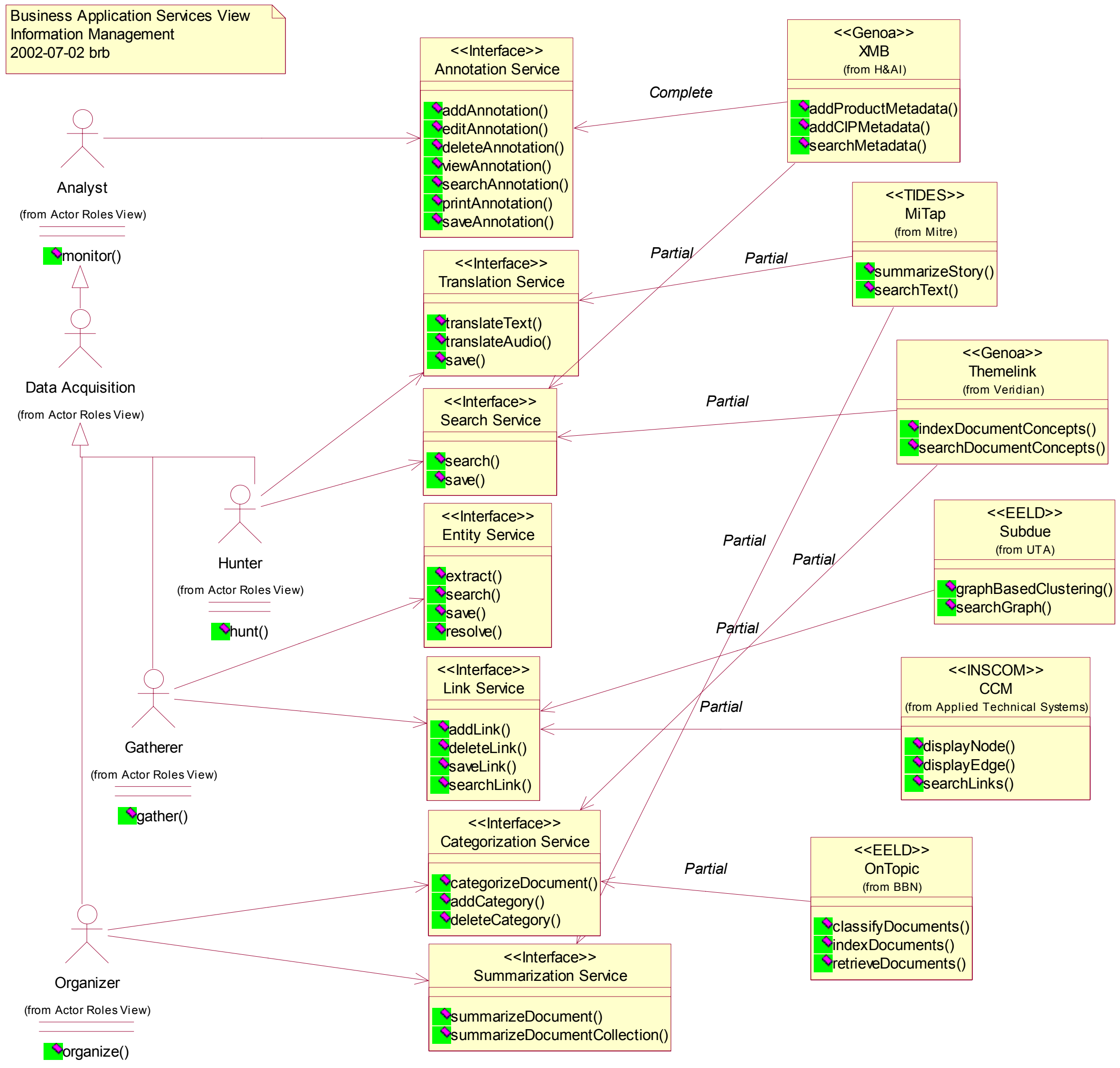
3.1.3.2.1 Collaboration Service

3.1.3.2.2 Security Service

3.1.3.2.3 Metadata Service

3.1.3.2.4 Repository Service

3.1.3.3 Information Management Business Application Services



3.1.3.3.1 Annotation Service

3.1.3.3.2 Translation Service

3.1.3.3.3 Search Service

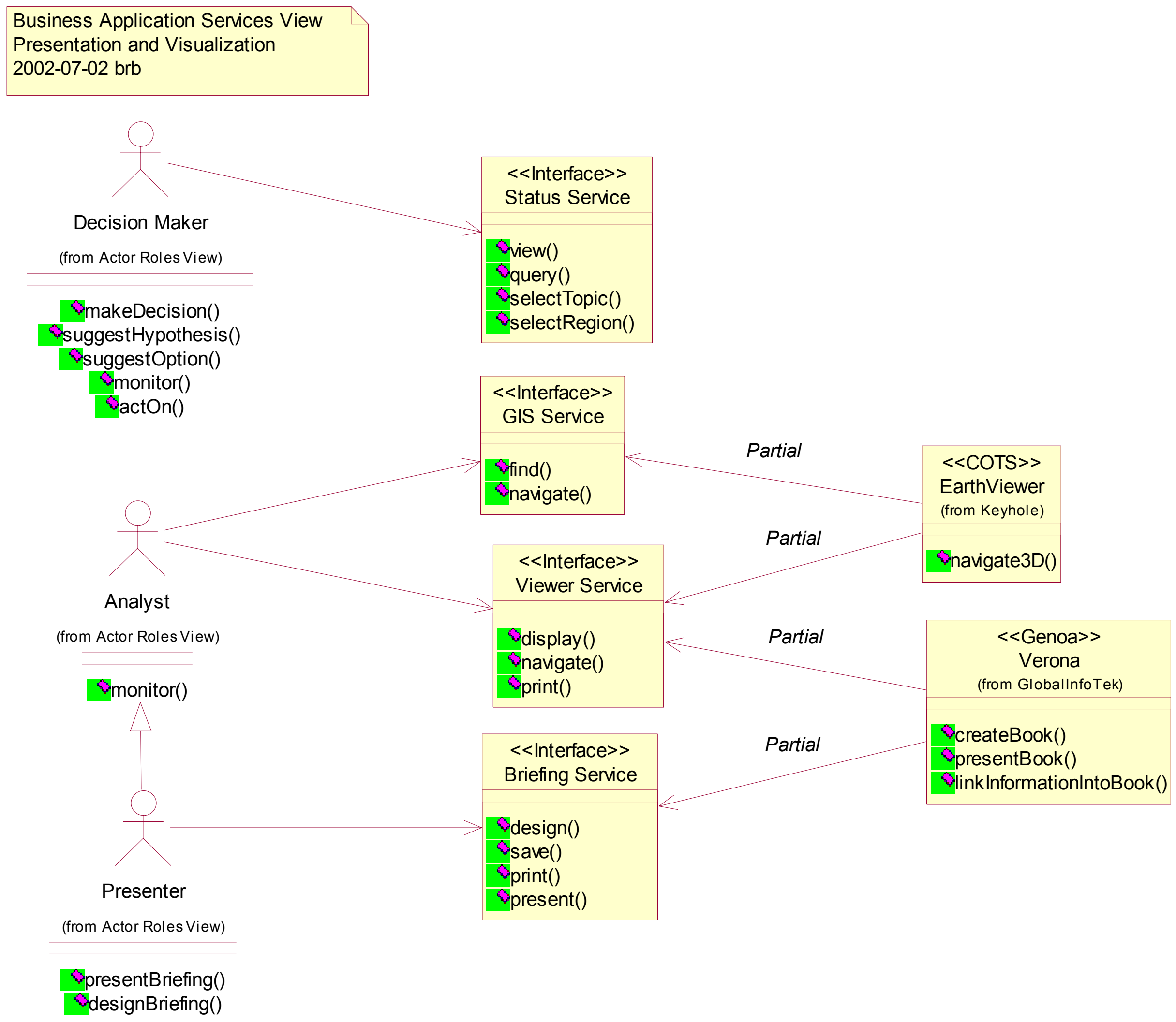
3.1.3.3.4 Entity Service

3.1.3.3.5 Link Service

3.1.3.3.6 Categorization Service

3.1.3.3.7 Summarization Service

3.1.3.4 Presentation and Visualization Business Application Services



3.1.3.4.1 Status Service

3.1.3.4.2 GIS Service

3.1.3.4.3 Viewer Service

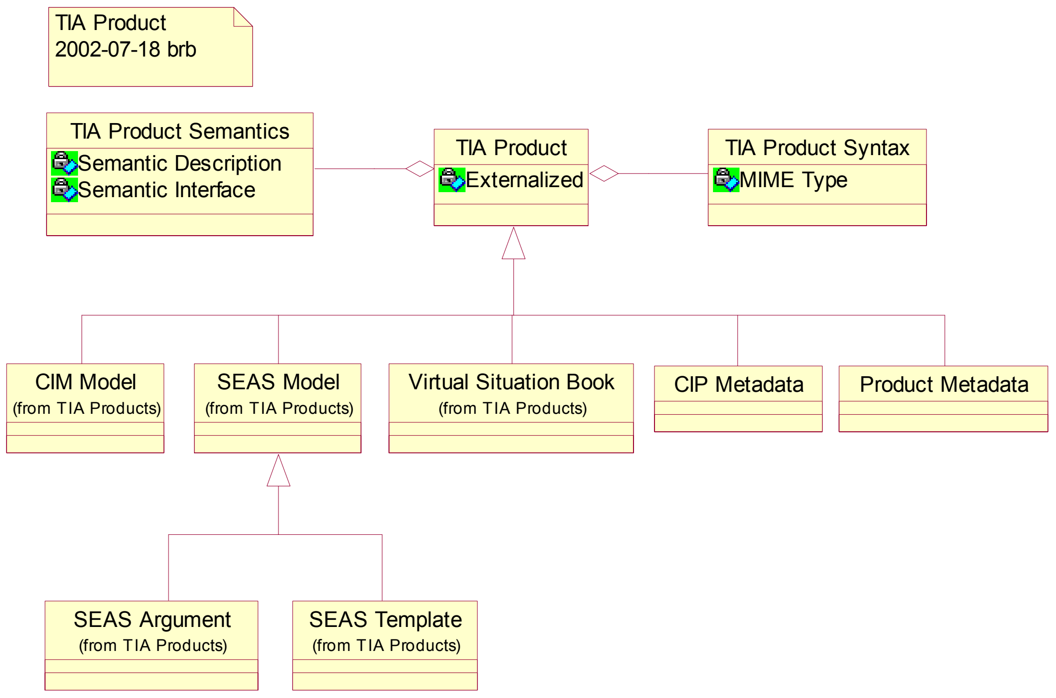
3.1.3.4.4 Briefing Service

3.1.4. Business Information View

The Business Information View contains classes representing the information products a user associates as inputs and outputs of the tools in the Business Application Services View. The products are defined in terms of a TIA Product (Section 3.1.4.1) and related TIA Product Format (Section 3.1.4.1.1).

3.1.4.1 TIA Product

The TIA Product represents the externalized information that a user associates as inputs and outputs. Externalized means that the "product" is in a format enabling it to be "passed-by-value", completely independently of the tool for which the product is an input and/or output. The TIA Product contains a description of the product format. The products shown in the diagram below represent those current used in the experimentation process.



3.1.4.1.1 CIM Model

The CIM model is used by Veridian's CIM (See Section 3.2.8.2.1).

3.1.4.1.2 SEAS Model

The SEAS Argument and Template are used by SRI's SEAS (See Section 3.2.8.2.2).

3.1.4.1.3 Virtual Situation Book

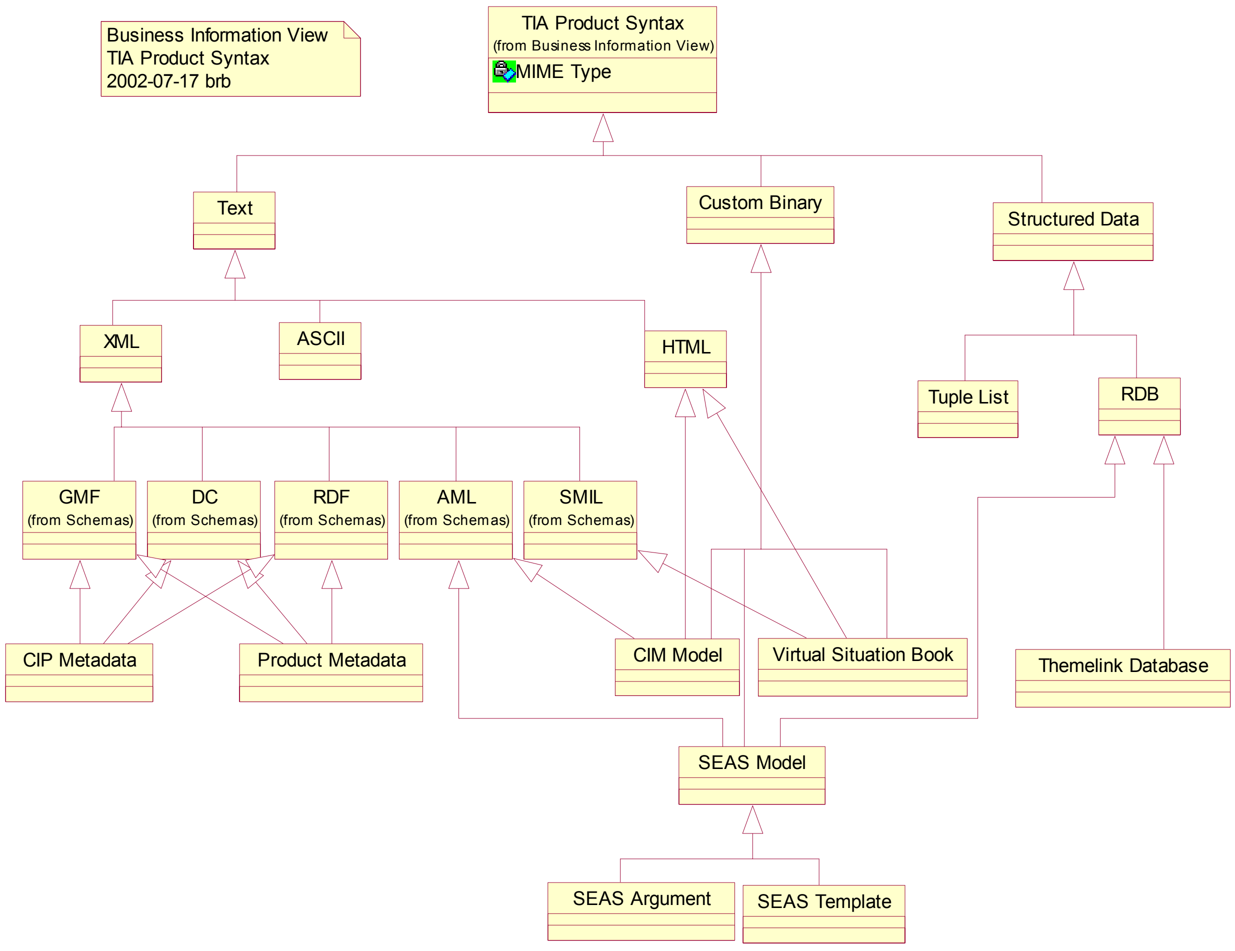
The Virtual Situation Book is used by GITI's Verona (See Section 3.2.8.2.5).

3.1.4.1.4 CIP and Product Metadata

CIP and Product Metadata may be created and managed using the XMB (See Section 3.2.8.2.6).

3.1.4.1.5 TIA Product Syntax

The TIA Product Syntax, shown in the next figure, describes the structure of TIA Information Products. Initially, this has been divided into three types: (1) Text, including structured and unstructured text, (2) Custom Binary, including all binary formats requiring a proprietary tool to access, and (3) Structured Data, including relational and other databases.



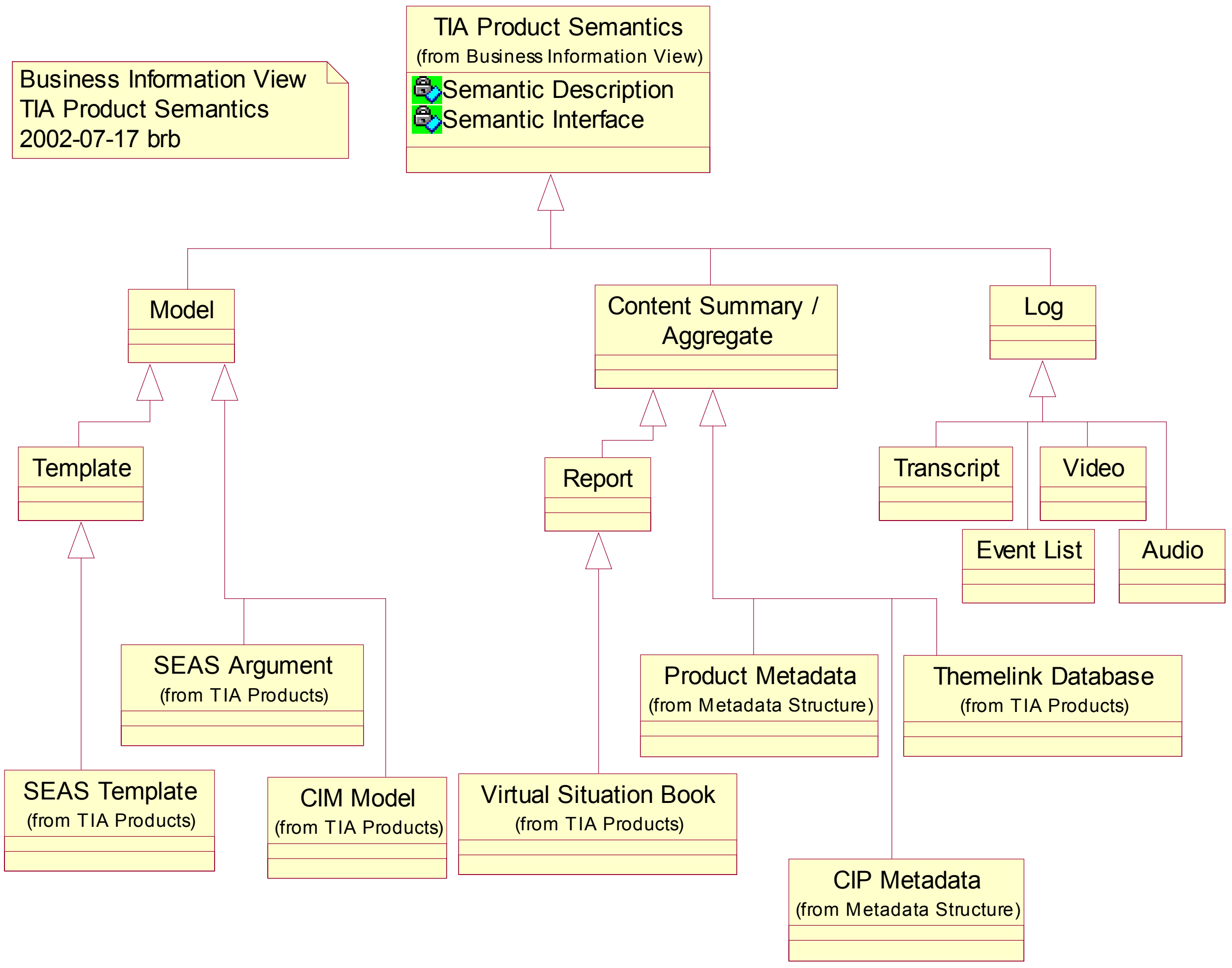
3.1.4.1.5.1 Relationship to the Information View

The TIA Product Syntax provides the linking between the Business Information View and the Information View (See Section 3.2.6) in the System model. Each type of product format may be associated with one or more Schemas from the Semantic View (See Section 3.2.6.1), e.g. Genoa Metadata Framework (GMF), Dublin Core (DC), Resource

Description Framework (RDF),Argument Markup Language (AML), and Streaming Multi-media Interchange Language (SMIL).

3.1.4.2 TIA Product Semantics

The TIA Product Semantics, shown in the next figure, represents the semantics of products from the perspective of the user. It is the semantic portion of the Business Information View. As the model develops, related taxonomies, ontologies, and other portions of the semantic description and interface (See Section 3.2.6.1.1) will be associated with each product.



3.1.4.3 Experiment Metrics

As experimentation is defined, metrics will be defined for each software application included in the experiment. They will be captured in the System Description and linked to the tool and information products that they are used to measure.

3.1.5. Experiment Deployment View

The Experiment Deployment view describes the mapping(s) of the software, identified in the Business Application Service View and used in Experiments, onto the hardware and reflects its distributed aspect. It is analogous to the Physical View of the Rational 4 + 1 View of System Architecture.

Each experiment can be considered a snapshot of the system architecture and business model as it exists during an experiment. It serves to record the implementation status and evolution of the system.

The Experiment Deployment View currently forward-references the existing experiment models contained in this document.

3.1.5.1 UML Deployment Diagram

See Section 4.1.3.1.

3.1.5.2 System Topology Diagram

See Section 4.1.3.2.

3.1.5.3 Mistral Experiment (22 May 2002)

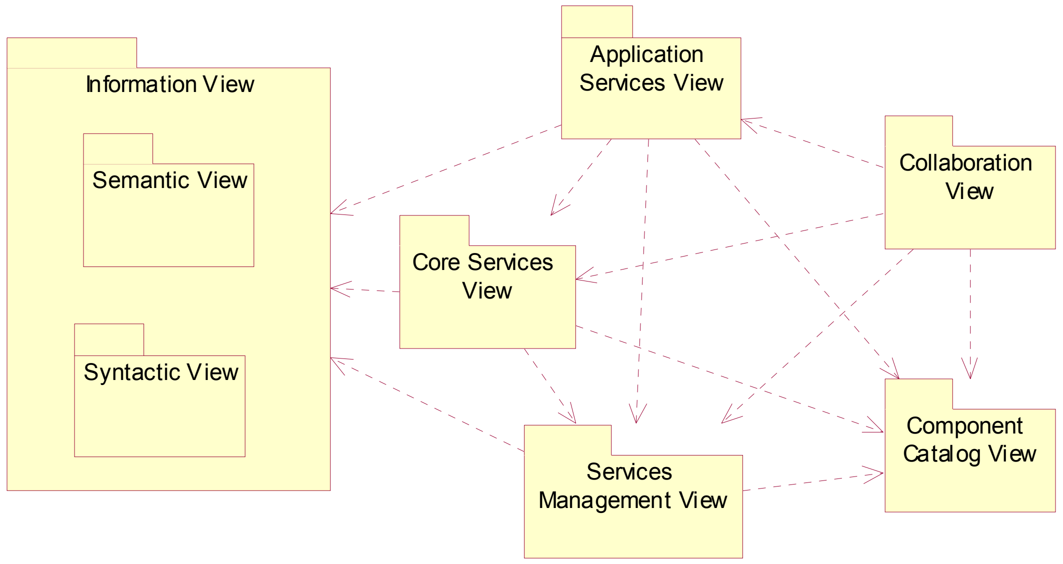
See Section 4.1.

3.1.5.4 Sirocco Experiment (August 2002)

See Section 4.2.

3.2. *Systems Model*

The Systems Model describes the arrangement, interaction, and interdependence of elements at a systems and platform level. It provides the technical systems implementation guidelines upon which engineering specifications are based and common building blocks are established.

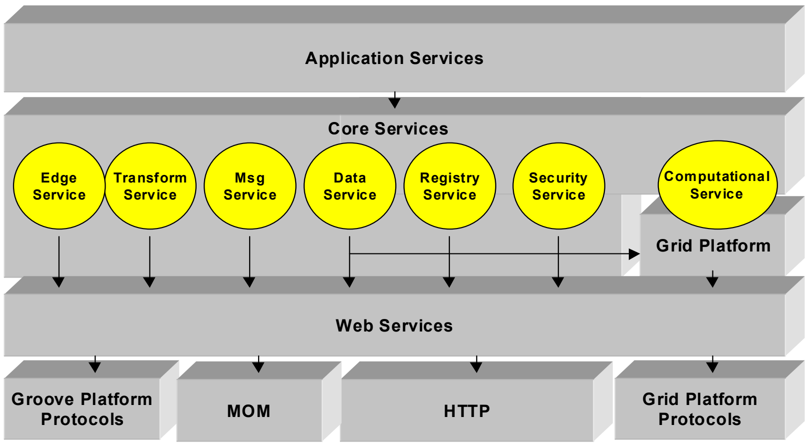


Views of the System Model

3.2.1. Introduction

As seen above, the Systems Model is based on a Service Oriented Architecture (SOA) and encompasses a set of interoperating Application Services that use core services and a common information model. These services will be accessed through an edge-based collaboration environment or a center-based portal and managed using a services management console. It is intended as an evolving work-in-progress to encompass the services "substrate" required by the TIA System to operate in a loosely coupled, highly cohesive system. The component view of the Systems Model, which will describe the application components provided by commercial off-the-shelf (COTS) or government off-the-shelf (GOTS) contractors, will be used to instantiate the functions in the systems model. The initial set of identified services will be described using a modeling process, which also can be applied to explain other services as necessary.

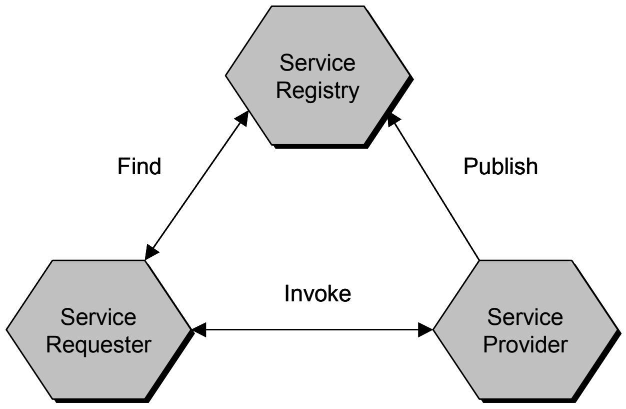
3.2.2. Layered Model Overview



The figure above depicts the Systems Model's critical functions in a layered format, which also identifies the various dependencies for each function. As shown, there are four identified platforms, at different layers, that will be required to enable the transport of the service protocols to fulfill the intended service: a Web Services platform, a Grid platform, the Groove Platform, and a Messaging Oriented Middleware (MOM) platform. The purpose of four interoperating platforms is that no one platform can support the complete set of transport or service functions required to enable all services. In addition, this approach will enable the incremental release of services until a potential candidate, such as the Open Grid Services Architecture (OGSA), is able to support all the necessary functions required for a complete, integrated system.

3.2.2.1 Service Oriented Architecture Overview

Web Services is a set of standards and techniques that represent a major paradigm shift in distributed computing from Distributed Object Architecture (DOA) to SOA. Under SOA, a set of network-accessible operations and associated resources are abstracted as a "service," which is described in a standard fashion, made available by publishing the service description to service registries, and found by the service requestor through querying the registry.



The TIA architecture will support Web Services that can be invoked using traditional static binding or dynamic invocation techniques. When employing the dynamic invocation of services, the service consumer will query the registry to find the service definition and bind to the service implementation at run-time. When using the static binding approach, service end-point information is known *a priori* and incorporated into the client code rather than accessing via registry lookup.

Web Services standards include the following:

- Simple Object Access Protocol (SOAP), which performs the low-level XML "plumbing" for transmitting web service calls across the wire;
- Web Service Definition Language (WSDL) is the language that defines the functional interfaces. In other words, a WSDL document (which is itself in XML) represents the official "contract" between the service consumers and providers. All TIA core services described in the later sections of this document are defined using WSDL.
- Universal Discovery, Description, and Integration (UDDI) is the standard for organizing and accessing the service registry, in the previous figure, service registry services as the "yellow" page of a set of Web Services, providing mechanisms for a service provider to publish its capabilities and for a user to find matching services.

Compared with traditional distributed computing technologies such as CORBA and DCE, Web Services a variety of significant advantages, including:

- **Maximum Interoperability**. Web Services standards are entirely based on XML and are programming language-, platform-, and programming model-neutral. For example, a Visual Basic client (written in a procedural model) on a Microsoft .NET server can readily invoke a Web Service hosted by a Java J2EE application server (written in an OO model) on a Red Hat Linux platform.
- **Loose Coupling**. Web Services standards focus on defining the functional *interfaces*, which represent the minimal understanding between service requestor and provider. Knowledge of the service provider is discovered dynamically from a service registry rather than statically coded in the client program.
- **Ubiquity**. Web Service calls are essentially XML messages over HTTP (or, potentially, other standard Internet protocols), which represent the "least common denominator" of network protocol stacks. This makes it easier to overcome firewall and infrastructure constraints. When it comes to cross-agency information sharing among different autonomous networks, Web Services is likely to be the only viable option.
- **Usability**. The same SOAP/XML/HTTP technologies also represent a low entry barrier for programmers, administrators, and users, making web services much easier to adopt and implement.

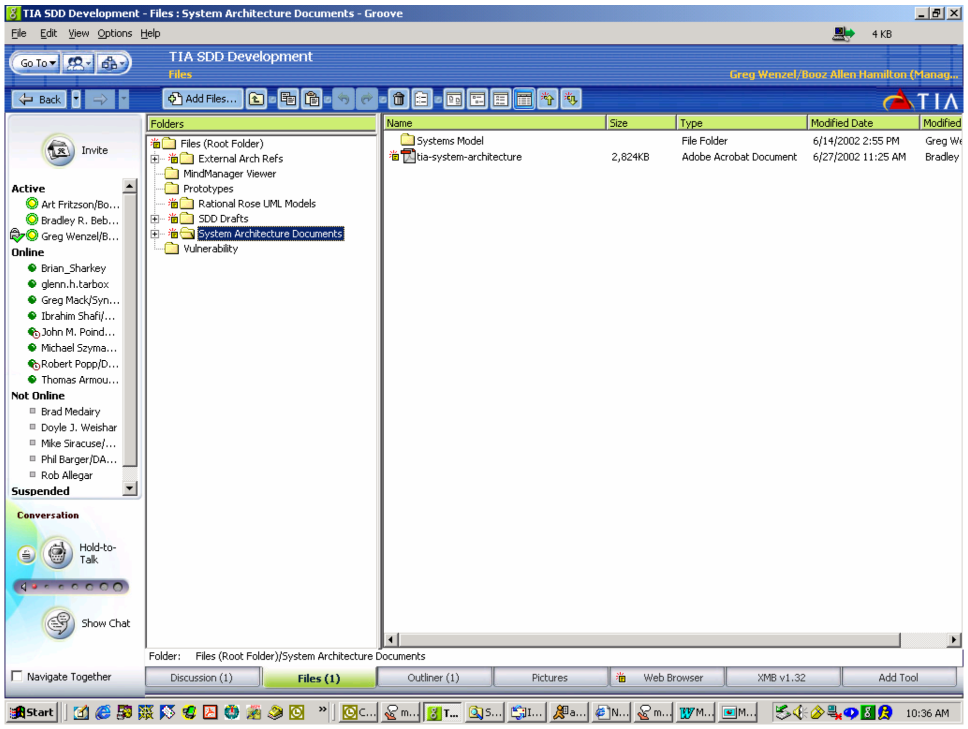
By laying its foundation on SOA and Web Services standards, the TIA system architecture is able to integrate a diverse set of core service capabilities, including security, messaging, and data transformation, into a unified yet highly scalable platform, as higher-level services may be composed using other services.

3.2.3. Collaboration View

The collaboration view contains the components that support the human interaction with the TIA system actors and between human actors. It contains the user interface that provides a common look, feel, and brand to which any counterterrorism analyst will become accustomed.

There are two main platforms for collaboration: edge and center. Furthermore, these platforms intersect with two collaborating methodologies: synchronous and asynchronous. Edge collaboration is a decentralized approach to collaboration that supports a peer-to-peer methodology in which each end-user is empowered with a full set of information replicated on their individual computers and is accessed through a rich heavy client-based user interface. Due to the replication of content to the edge, it is best used for short duration, synchronous collaboration with a smaller member set in the collaboration space. In the case of center-based collaboration, information is stored centrally on a server and is accessed typically using a web browser. Center-based collaboration is best used in longer duration asynchronous collaboration situations with a larger set of members in the space. Note, however, that there are areas where edge-based collaboration platforms can support asynchronous collaboration and center-based collaboration can support synchronous collaboration.

It has been envisioned that TIA will employ a process using a combination of these two collaboration platforms and methodologies. All Application Services will be accessed using one of two collaboration user interfaces. For edge-based collaboration, a peer-to-peer product such as Groove will be used to support the decentralized collaboration activities. Application Services requiring an edge-based approach will be integrated as a tool within the Groove platform and will provide a rich user interface based on the Groove UI as in the figure below.


TIA SDD Development - Files : System Architecture Documents - Groove

TIA SDD Development
Files
Greg Wenzel/Booz Allen Hamilton (Manag...

Active
- Art Fritzson/Bo...
- Bradley R. Beb...
- Greg Wenzel/B...

Online
- Brian_Sharkey
- glenn.h.tarbox
- Greg Mack/Syn...
- Ibrahim Shafi/...
- John M. Poind...
- Michael Szyma...
- Robert Popp/D...
- Thomas Armou...

Not Online
- Brad Medairy
- Doyle J. Weishar
- Mike Siracuse/...
- Phil Barger/DA...
- Rob Allegar

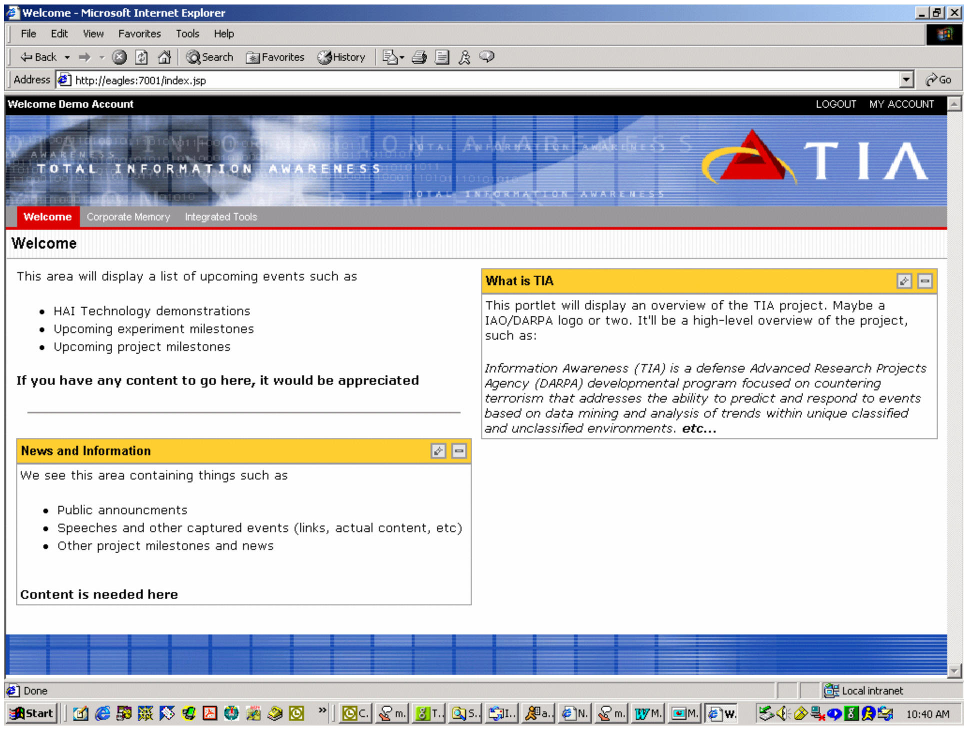
Suspended

Folders
- Files (Root Folder)
  - External Arch Refs
  - MindManager Viewer
  - Prototypes
  - Rational Rose UML Models
  - SDD Drafts
  - System Architecture Documents
  - Vulnerability

Name	Size	Type	Modified Date	Modified
Systems Model		File Folder	6/14/2002 2:55 PM	Greg We
tia-system-architecture	2,824KB	Adobe Acrobat Document	6/27/2002 11:25 AM	Bradley

Folder: Files (Root Folder)/System Architecture Documents

For the center-based collaboration user interface, a TIA portal will be established to organize the UI using a thin client running within a browser. Collaboration "portlets" will be developed for Application Services that are best suited to running in a center-based thin client mode.


Welcome - Microsoft Internet Explorer

Address http://eagles:7001/index.jsp

Welcome Demo Account — LOGOUT MY ACCOUNT

TOTAL INFORMATION AWARENESS — TIA

Welcome | Corporate Memory | Integrated Tools

Welcome

This area will display a list of upcoming events such as

- HAI Technology demonstrations
- Upcoming experiment milestones
- Upcoming project milestones

If you have any content to go here, it would be appreciated

News and Information

We see this area containing things such as

- Public announcments
- Speeches and other captured events (links, actual content, etc)
- Other project milestones and news

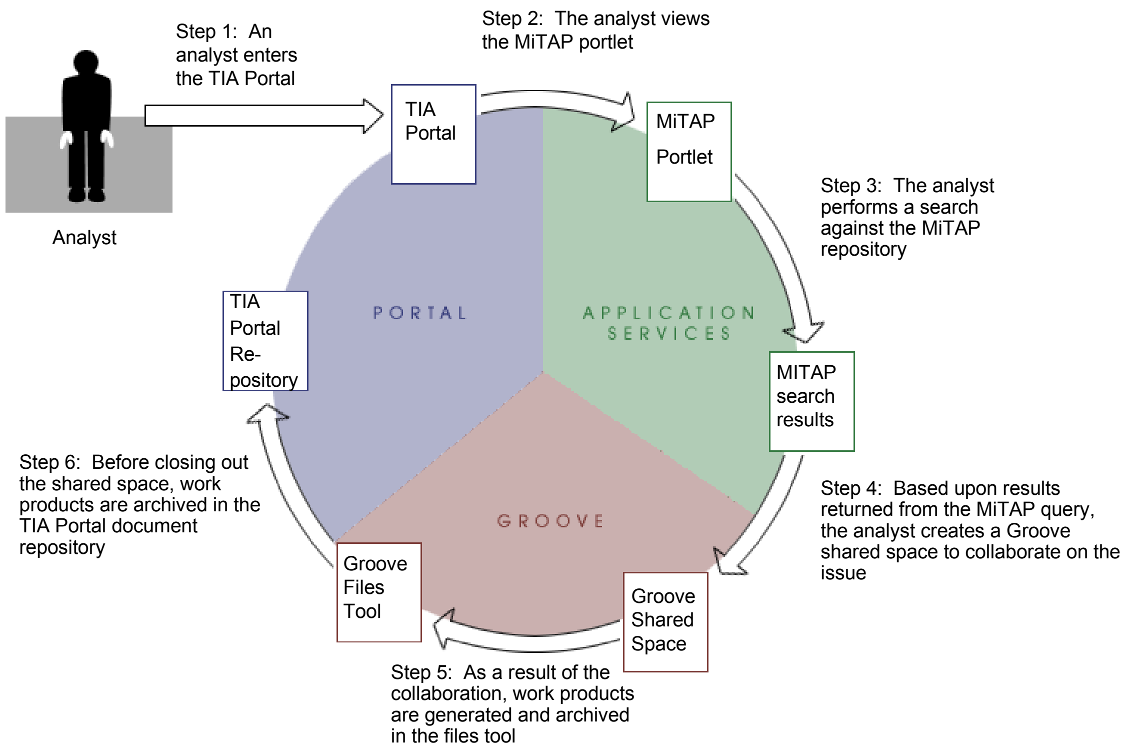
Content is needed here

What is TIA

This portlet will display an overview of the TIA project. Maybe a IAO/DARPA logo or two. It'll be a high-level overview of the project, such as:

Information Awareness (TIA) is a defense Advanced Research Projects Agency (DARPA) developmental program focused on countering terrorism that addresses the ability to predict and respond to events based on data mining and analysis of trends within unique classified and unclassified environments. ***etc...***

Finally, the TIA architecture will support interoperability between these two collaborating platforms. Using the core services described below, TIA will support a full round-trip collaborating environment between the edge- and center-based platforms. The intent for round-trip collaboration, shown in the next figure, is so that the user does not have to manually copy data between these environments.


Step 1: An analyst enters the TIA Portal
Analyst
Step 2: The analyst views the MiTAP portlet
TIA Portal
MiTAP Portlet
Step 3: The analyst performs a search against the MiTAP repository
PORTAL
APPLICATION SERVICES
MITAP search results
TIA Portal Re-pository
Step 4: Based upon results returned from the MiTAP query, the analyst creates a Groove shared space to collaborate on the issue
GROOVE
Step 6: Before closing out the shared space, work products are archived in the TIA Portal document repository
Groove Files Tool
Groove Shared Space
Step 5: As a result of the collaboration, work products are generated and archived in the files tool

3.2.4. Application Services View

Application Services are higher order services built upon TIA core services and possibly other higher order services. Application Services encapsulate an application's business logic and/or algorithms and expose this information as a service for other applications to consume. This approach will enable a "chaining" effect, that is, that incremental logic of differing applications can be built successively on top of other services, which, in turn, will create a more powerful tool. Typical candidates that could be used for Application Services are applications or algorithms that do not have a user interface and run via command line. Another candidate are those cases where the application logic can be separated from the user interface, such as for evidence extraction or link discovery, making it available to external applications as a "black box."

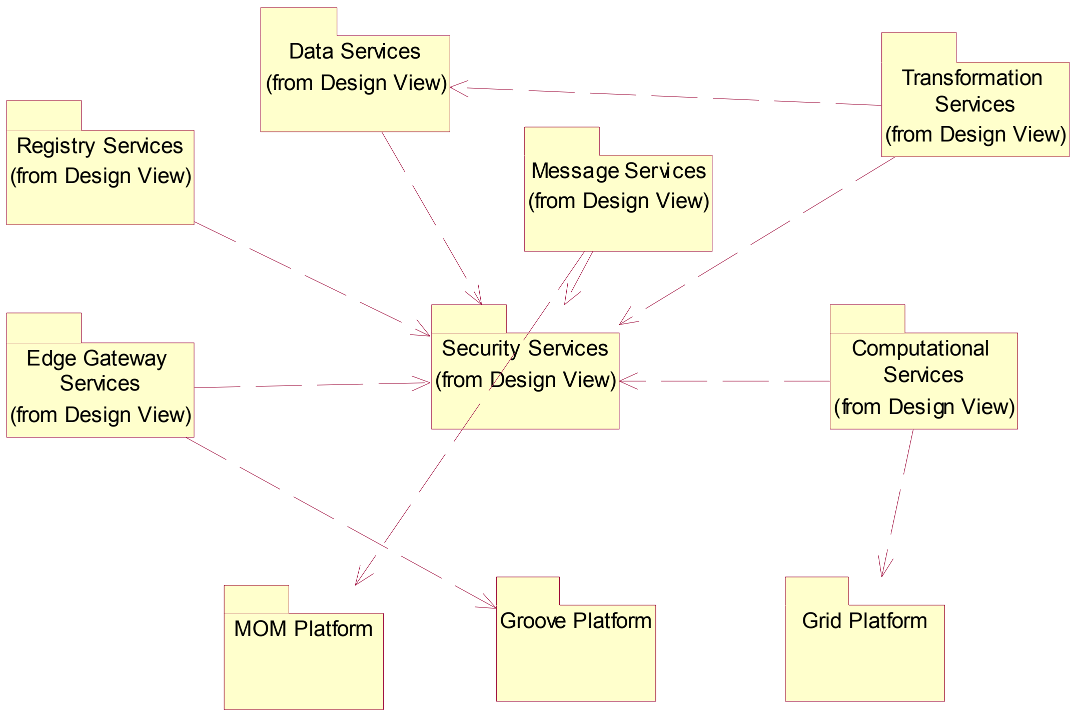
This section serves as the placeholder for the elaboration of the Application Services, which will evolve and be defined throughout the evolution of the TIA system. Once a service is identified during TIA experiments, Use Cases, Interface Description, and

WSDL Interface will be created describing the service. These artifacts will be developed for each of these higher order services.

3.2.5. Core Services View

The following sections will describe the fundamental Core Services required of the TIA System. These sections will be broken down into their constituent parts by identifying the System Use Cases for each service, the Class Diagrams describing their external interfaces to the consumers and the high level underlying design behind the interface. A Web Services Definition Language (WSDL) [4] interface for each service is provided as an Appendix.

The figure below depicts the logical view of the Core Services. This diagram describes the inter-dependencies of the services to provide the security and administration functions required of an interoperating system of services

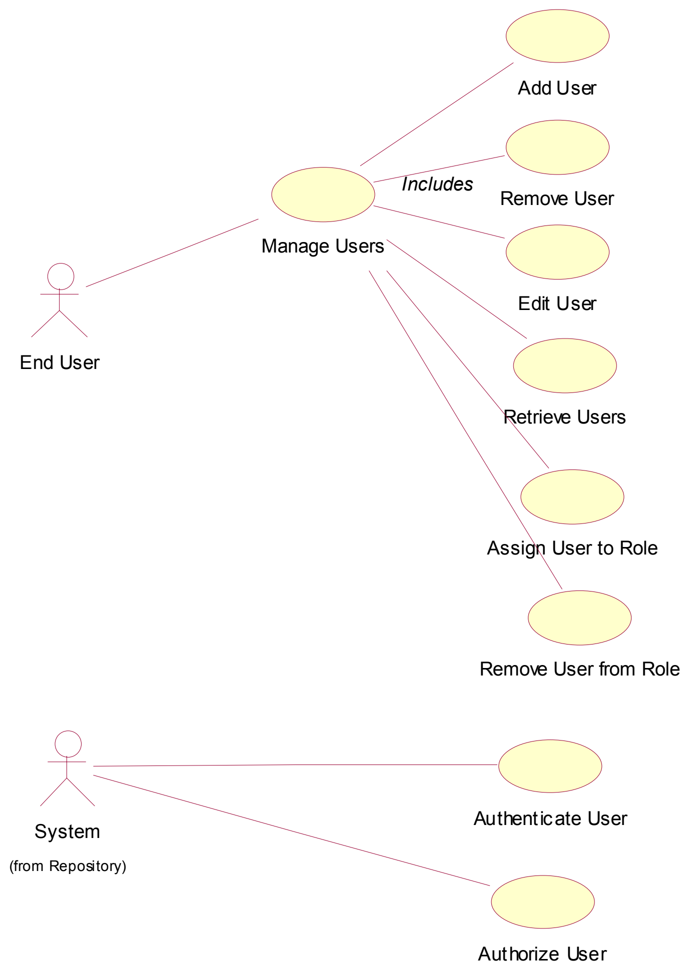


3.2.5.1 Security Services

Central to an integrated system, Security Services will provide the necessary functions to enable applications to authenticate and authorize consumers of each service to operate within a central security framework. Security Services will provide a centralized administration scheme to manage users and their privileges across the consumers of the services, realizing a single sign-on across the systems.

Currently, native security provisions in the existing Web Services standards are extremely lacking. Core features of a base security framework including authentication and authorization have not been sufficiently addressed. The high-level design in this section represents a baseline custom solution that will provide interim support for authentication and authorization. We anticipate that the emergence of new standards for Web Service security [6] introduced by IBM and Microsoft (WS-Security, WS-Authentication, and WS-Policy) will supersede the design reflected below.

3.2.5.1.1 Security Services System Use Cases



The Use Case diagram above highlights the initial set of functionality proposed for security services. The core set of functionality includes the management of user accounts along with security authentication and authorization.

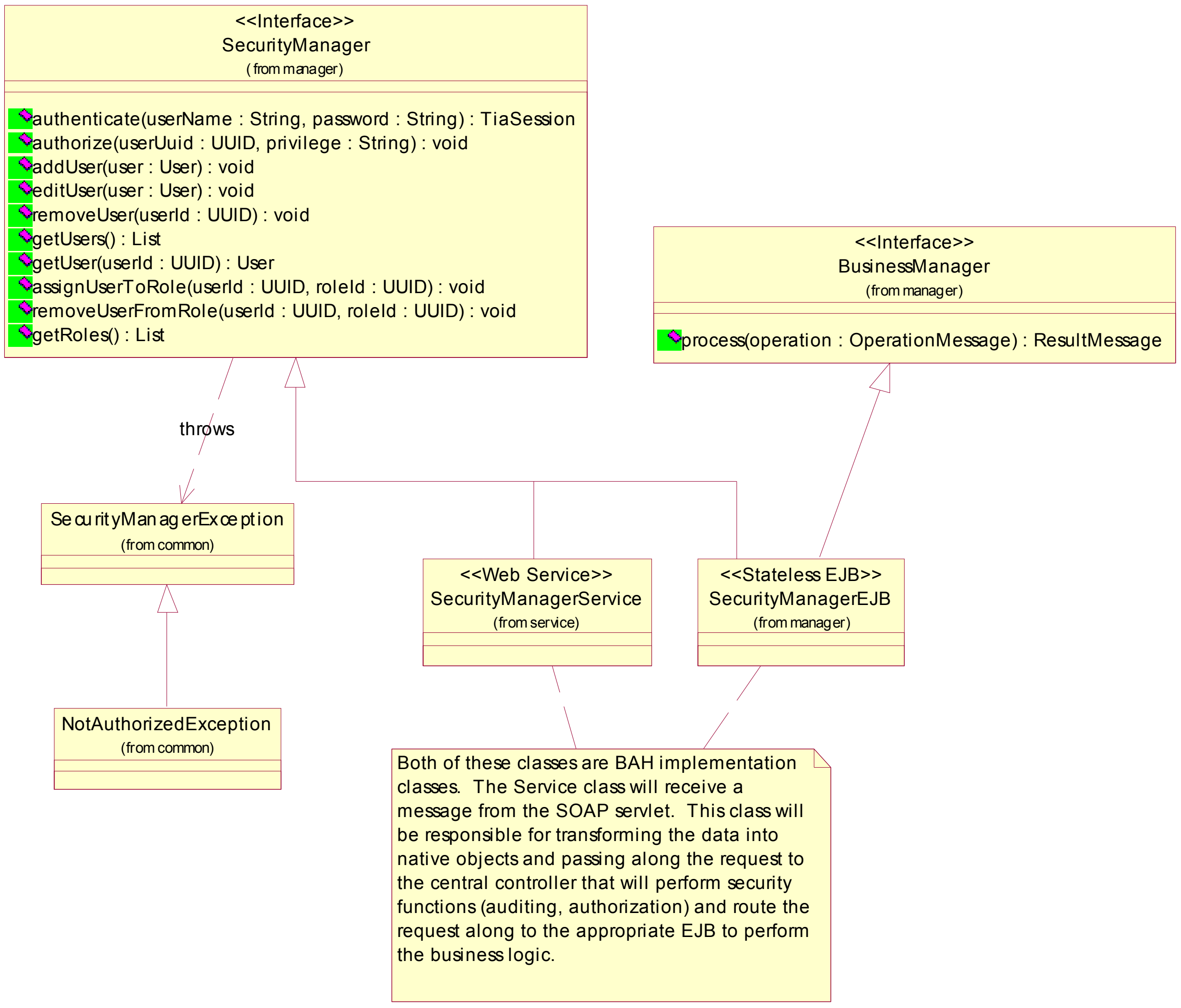
The Manager Users Use Cases reflect the ability to add new user accounts to the system, update user metadata, remove a user from the system, retrieve a list of users from the system, and manage a user's role assignments. User information likely will be persisted in an underlying store, such as LDAP or a relational database.

Authentication of a user is intended to restrict access of Core Service and Application Service components to validated users of the TIA system. If a user has not been authenticated with the Security Service, the end-user request will be denied.

Authorization of a user request involves the validation of a user's credentials to ensure he or she has the proper role assignment and privilege to invoke the service operation. If a user is not authorized to perform a request, he or she will receive a security exception and the operation will be aborted.

3.2.5.1.2 Security Services Interface

Based upon the Use Case diagram defined above, the following class diagram has been defined to satisfy the base requirements:


<<Interface>>
SecurityManager
(from manager)
authenticate(userName : String, password : String) : TiaSession
authorize(userUuid : UUID, privilege : String) : void
addUser(user : User) : void
editUser(user : User) : void
removeUser(userId : UUID) : void
getUsers() : List
getUser(userId : UUID) : User
assignUserToRole(userId : UUID, roleId : UUID) : void
removeUserFromRole(userId : UUID, roleId : UUID) : void
getRoles() : List
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
throws
SecurityManagerException
(from common)
<<Web Service>>
SecurityManagerService
(from service)
<<Stateless EJB>>
SecurityManagerEJB
(from manager)
NotAuthorizedException
(from common)

Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.

The class diagram above depicts the initial high-level design representation of the TIA Data Services. An interface called "SecurityManager" has been created to serve as the service view to the external world. The SecurityManager interface exposes key methods that enable both user account management and base Security Services, including authentication and authorization. The SecurityManager interface will generate the WSDL service description that developers will use to generate programs that communicate with the service.

The SecurityManager is required to throw an exception called "SecurityManagerException" if a request cannot be fulfilled. The SecurityManagerException will map to a fault as specified in Section 4.3.6 of the JAX-RPC .8 specification[1].

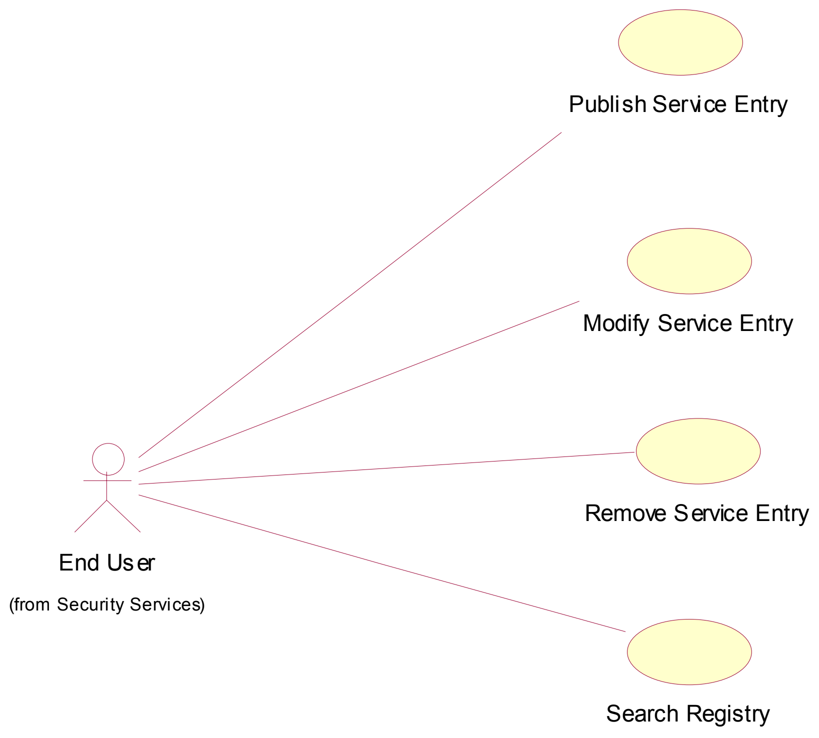
Similar to the implementation patterns applied in other service components, two implementation-specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. Initially, the SecurityManagerService will implement the SecurityManager interface and serve as the entry point for incoming Web Service requests. The SecurityManagerService class will likely perform any necessary object assembly and forward the request to a central control for security, logging, and routing. The SecurityEJB class will likely be implemented as a stateless Enterprise Java Bean (EJB) and perform the necessary business logic to service the request. The diagram above reflects the SecurityEJB implementing a BusinessManager interface. The BusinessManager interface will enable the SecurityManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller where other services, such as security and logging, will occur.

3.2.5.2 Registry Services

The TIA Registry Services will provide a mechanism to enable the location of service providers and the retrieval of service description documents for the TIA system. Registry Services will take advantage of a central registry server that will be enabled through the Universal Description, Discovery, and Integration (UDDI) specification [5].

Initially, software requesters will manually search registries to discover service descriptions and use traditional static binding techniques to employ the service. In the future, software programs will use the registry service to dynamically discover and bind services at runtime.

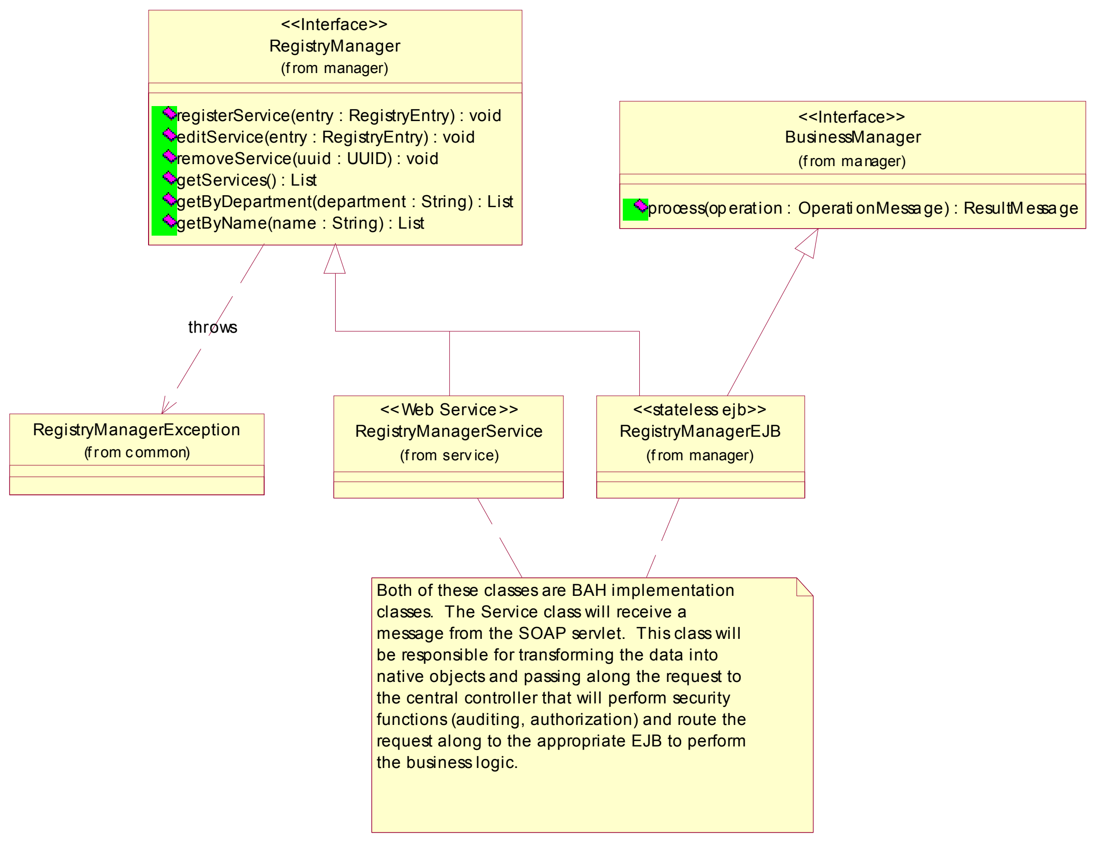
3.2.5.2.1 Registry Services System Use Cases



The Use Case diagram presented above highlights the initial set of functionality proposed for registry services. The core set of functionality includes publishing, modifying, and removing a service entry and searching the registry.

Publishing a service entry involves creating the service description in the UDDI registry. A service description will comply with the WSDL, which is based on XML and used to describe a Web Service. [4]. The WSDL service description will contain the operations supported by the service, the network endpoint for the service, and the protocol binding of the service. Modifying a service entry involves the update of the WSDL service description or any of the metadata capture about the service description. Removing a service entry involves deleting the WSDL service description and associated metadata from the UDDI repository. Once information has been removed from the repository, it cannot be recovered. Searching the registry will enable developers and application programs to locate service descriptions based upon specified search criteria.

3.2.5.2.2 Registry Services Interface



<<Interface>>
RegistryManager
(from manager)
registerService(entry : RegistryEntry) : void
editService(entry : RegistryEntry) : void
removeService(uuid : UUID) : void
getServices() : List
getByDepartment(department : String) : List
getByName(name : String) : List
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
throws
RegistryManagerException
(from common)
<<Web Service>>
RegistryManagerService
(from service)
<<stateless ejb>>
RegistryManagerEJB
(from manager)
Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.

The class diagram above depicts the initial high-level design representation of the TIA Registry Services. An interface called "RegistryManager" has been created to serve as the service view to the external world. The RegistryManager interface exposes key methods to enable basic Create/Read/Update/Delete (CRUD) functionality. The registry manager interface will be used to generate the WSDL service description that will be used by developers to develop programs that communicate with the service.

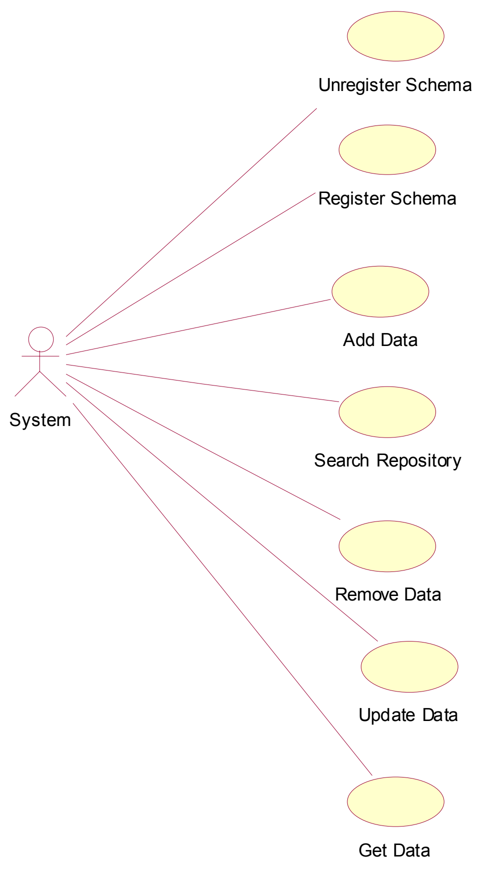
The RegistryManager is required to throw an exception called "RegistryManagerException" in the event a request cannot be fulfilled. The RegistryManagerException will map to a fault as specified in Section 4.3.6 of the JAX-RPC .8 specification.

Two implementation-specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. The initial intent of the RegistryManagerService is to implement the RegistryManager interface and serve as the entry point for incoming web service requests. The RegistryManagerService class will likely perform any necessary object assembly and forward the request to a central control where security, logging, and routing will occur. The RegistryManagerEJB class will likely be implemented as a stateless Enterprise Java Bean (EJB) and perform the necessary business logic to service the request. The diagram above reflects the RegistryManagerEJB implementing a BusinessManager interface. The BusinessManager interface will enable the RegistryManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller where services such as security and logging will occur.

3.2.5.3 Data Services

Data Services provides a loosely coupled access to local and remote data sources. In contrast to a standard enterprise (local) approach using JDBC or ODBC access to a data source, Data Services will extend the data across a WAN and will enable the dynamic binding of data resources to enable an aggregation of data again creating a “chaining” effect providing a richer set of information (i.e., knowledge) rather than offering simple raw data sources. Additionally, Data Services will be dependent on the Security Services described earlier, thus adding security and access control over different aspects of the data based on the privileges of the consumer role. This will provide users with different privileges—different “views”—of the same data set facilitating discretionary access control (DAC).

3.2.5.3.1 Data Services System Use Cases



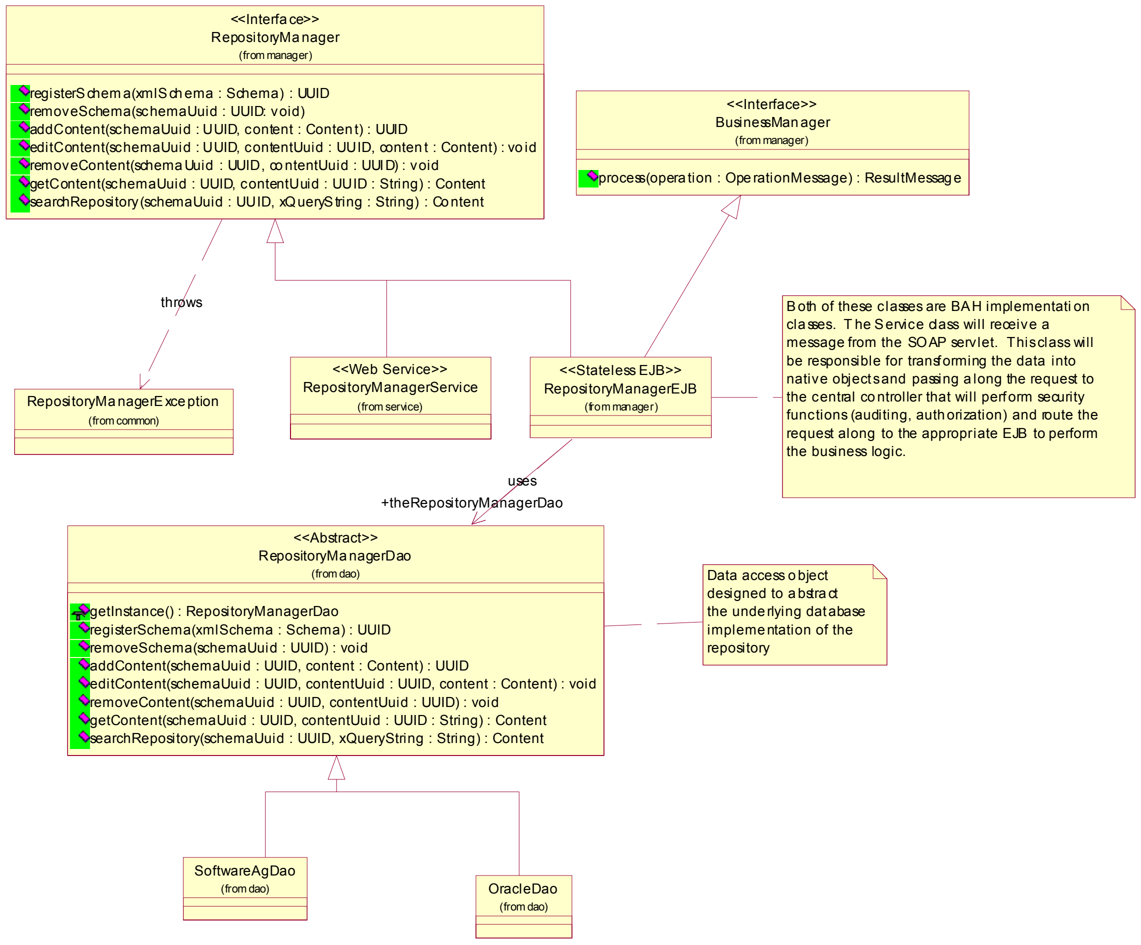
The Use Case diagram depicted above highlights the initial set of functionality proposed for Data Services. The core set of functionality includes managing schemas, managing data, and searching the repository. Additional Use Cases will be added for domain-specific data access as requirements are identified during experimentation.

Management of schemas involves the Register Schema and Unregister Schemas Use Cases. Registering a schema enables external applications and administrators to register a valid XML schema with the repository manager. The registered schema will validate incoming XML documents to be stored in the repository. The Unregister Schema Use Case removes the registered schema from the system. After the schema has been unregistered, new requests that coming in to add XML data will be unable to validate with the repository.

The management of data encompasses the Add Data, Update Data, Remove Data, and Get Data Use Cases. When adding data to the system, the incoming XML document will be validated against known XML schemas that have been registered with the repository. If the incoming XML document does not validate against a known schema, the add data request can be denied.

The Searching Repository use case encompasses a key component of the repository and data services, information retrieval. The current vision is to provide an approach for queries to be passed into the Repository Service where they will be executed against stored data and the results returned. Since the initial intent of the repository is to provide a store for XML-based data, the queries will likely be in the form of XPath or Query statements. Data can also be retrieved by querying based on the Universally Unique Identifier (UUID) that is assigned when data is entered into the repository.

3.2.5.3.2 Data Services Interface


<<Interface>>
RepositoryManager
(from manager)
registerSchema(xmlSchema : Schema) : UUID
removeSchema(schemaUuid : UUID: void)
addContent(schemaUuid : UUID, content : Content) : UUID
editContent(schemaUuid : UUID, contentUuid : UUID, content : Content) : void
removeContent(schemaUuid : UUID, contentUuid : UUID) : void
getContent(schemaUuid : UUID, contentUuid : UUID : String) : Content
searchRepository(schemaUuid : UUID, xQueryString : String) : Content
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
throws
RepositoryManagerException
(from common)
<<Web Service>>
RepositoryManagerService
(from service)
<<Stateless EJB>>
RepositoryManagerEJB
(from manager)
Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.
uses
+theRepositoryManagerDao
<<Abstract>>
RepositoryManagerDao
(from dao)
getInstance() : RepositoryManagerDao
registerSchema(xmlSchema : Schema) : UUID
removeSchema(schemaUuid : UUID) : void
addContent(schemaUuid : UUID, content : Content) : UUID
editContent(schemaUuid : UUID, contentUuid : UUID, content : Content) : void
removeContent(schemaUuid : UUID, contentUuid : UUID) : void
getContent(schemaUuid : UUID, contentUuid : UUID : String) : Content
searchRepository(schemaUuid : UUID, xQueryString : String) : Content
Data access object designed to abstract the underlying database implementation of the repository
SoftwareAgDao
(from dao)
OracleDao
(from dao)

The class diagram above depicts the initial high-level design representation of the TIA Data Services. An interface called "RepositoryManager" has been created to serve as the service view to the external world. The RepositoryManager interface exposes key methods to enable the registration and management of schemas along with the ability Create/Read/Update/Delete/Search data stored against the schema. The RepositoryManager interface will be used to generate the WSDL service description that will be used by developers to develop programs that communicate with the service.

The RepositoryManager Manager is required to throw an exception called "RepositoryManager ManagerException" in the event a request cannot be fulfilled. The RepositoryManagerException will map to a fault as specified in Section 4.3.6 of the JAX-RPC 8 specification [1].

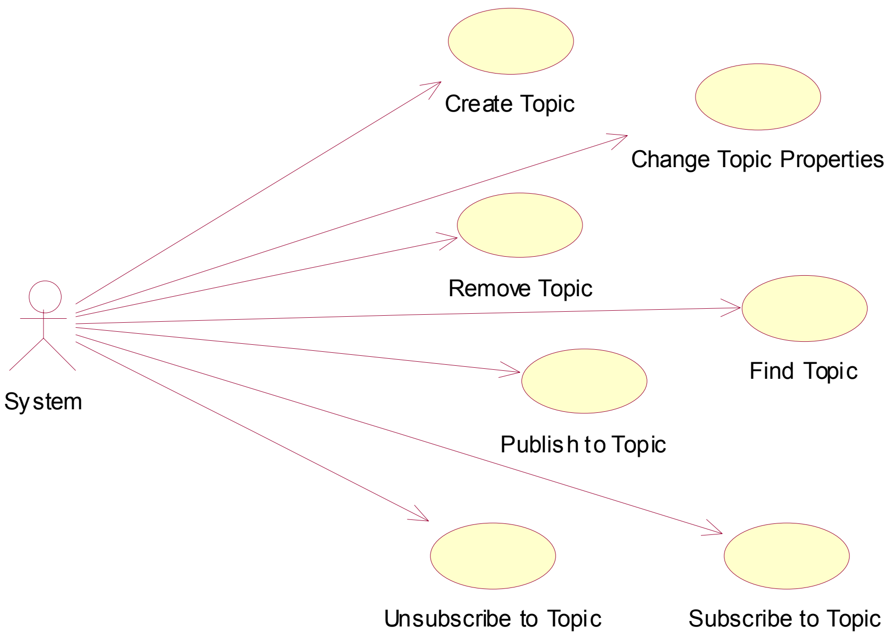
Similar to the implementation patterns applied in other service components, two implementation specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. The RepositoryManagerService is initially intended to implement the RepositoryManager interface and serve as the entry point for incoming Web Service requests. The RepositoryManagerService class will likely perform any necessary object assembly and forward the request to a central control for security, logging, and routing. The RepositoryEJB class will likely be implemented as a stateless Enterprise Java Bean (EJB) and perform the necessary business logic to service the request. The diagram above reflects the RepositoryEJB implementing a BusinessManager interface. The BusinessManager interface will enable the RepositoryManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller where services such as security and logging will occur.

A key design goal of the Repository Manager Service is to hide the underlying implementation of the actual data repository. Given the dynamic nature of the database market, the product we choose today will likely be replaced with a different, more robust solution in the future. To accomplish this design goal, we have applied the Data Access Object (DAO) design pattern. In the figure above, the RepositoryManagerDAO is an abstract base class that implements the singleton pattern. When the getInstance() operation is invoked on the RepositoryManagerDAO, the DAO class will make a decision, based on the environment in which it deployed, on what DAO implementation class to instantiate. The diagram currently represents DAO implementation classes for Oracle and Software AG, however, this can easily be expanded to support virtually any repository. The RepositoryManagerEJB will program to the RepositoryManagerDAO interface and will not be effected by "plugging in" a new implementation class.

3.2.5.4 Messaging Services

Messaging Services provides the asynchronous approach to data and computing intensive service requests. The Messaging Services allows multiple providers and consumers to subscribe to a messaging topic as a single interface rather than creating multiple sets of point-to-point connections. Furthermore, the asynchronous nature allows consumers to continue processing other activities and to be notified when their requests are completed. A callback mechanism will be put in place that will, upon notification, automatically invoke a function on the consumer to process the service results.

3.2.5.4.1 *Messaging Services System Use Cases*

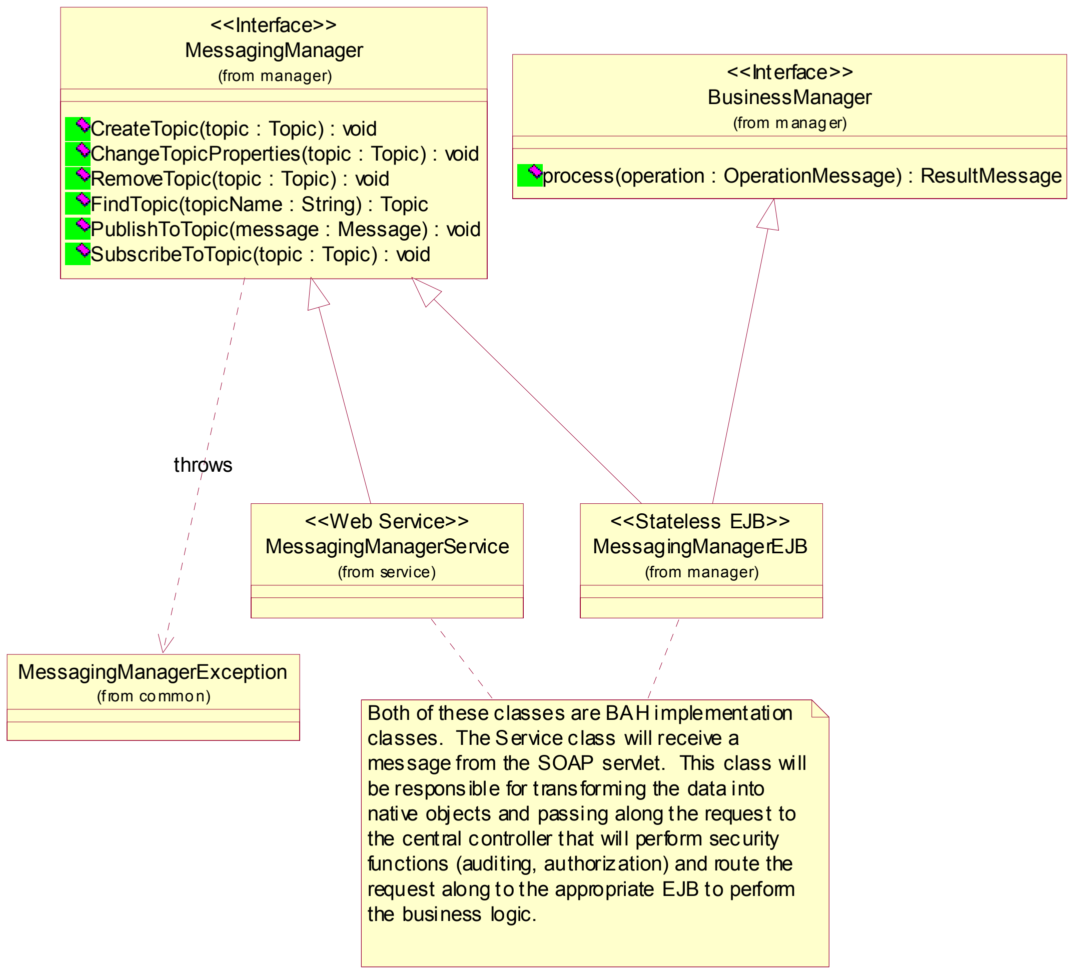


The Use Case diagram depicted above highlights the initial set of functionality proposed for messaging services. The core set of functionality includes managing topics and message retrieval.

Managing topics involves the creation, modification, and removal of a topic. Once a topic has been removed from the system, messages can no longer be published to the topic and any connections from active subscribers will be terminated.

Message retrieval involves systems locating topics they wish to subscribe to and, once the topic has been discovered, subscribing to the topic. When a system subscribes to a topic, it will receive all messages that are published to the topic. Published messages will likely be in the form of XML; other message formats include standards text messages and binary messages.

3.2.5.4.2 Messaging Services Interface


<<Interface>>
MessagingManager
(from manager)
CreateTopic(topic : Topic) : void
ChangeTopicProperties(topic : Topic) : void
RemoveTopic(topic : Topic) : void
FindTopic(topicName : String) : Topic
PublishToTopic(message : Message) : void
SubscribeToTopic(topic : Topic) : void
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
throws
<<Web Service>>
MessagingManagerService
(from service)
<<Stateless EJB>>
MessagingManagerEJB
(from manager)
MessagingManagerException
(from common)

Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.

The class diagram above depicts the initial high-level design representation of the TIA Messaging Services. An interface called “MessagingManager” has been created to serve as the service view to the external world. The MessagingManager interface exposes key methods to enable the management of messaging topics and the subscription and publishing to those topics. The MessagingManager interface will be used to generate the WSDL service description that will be used by developers to develop programs that communicate with the service.

The MessagingManager is required to throw an exception called “MessagingManagerException” in the event a request cannot be fulfilled. The

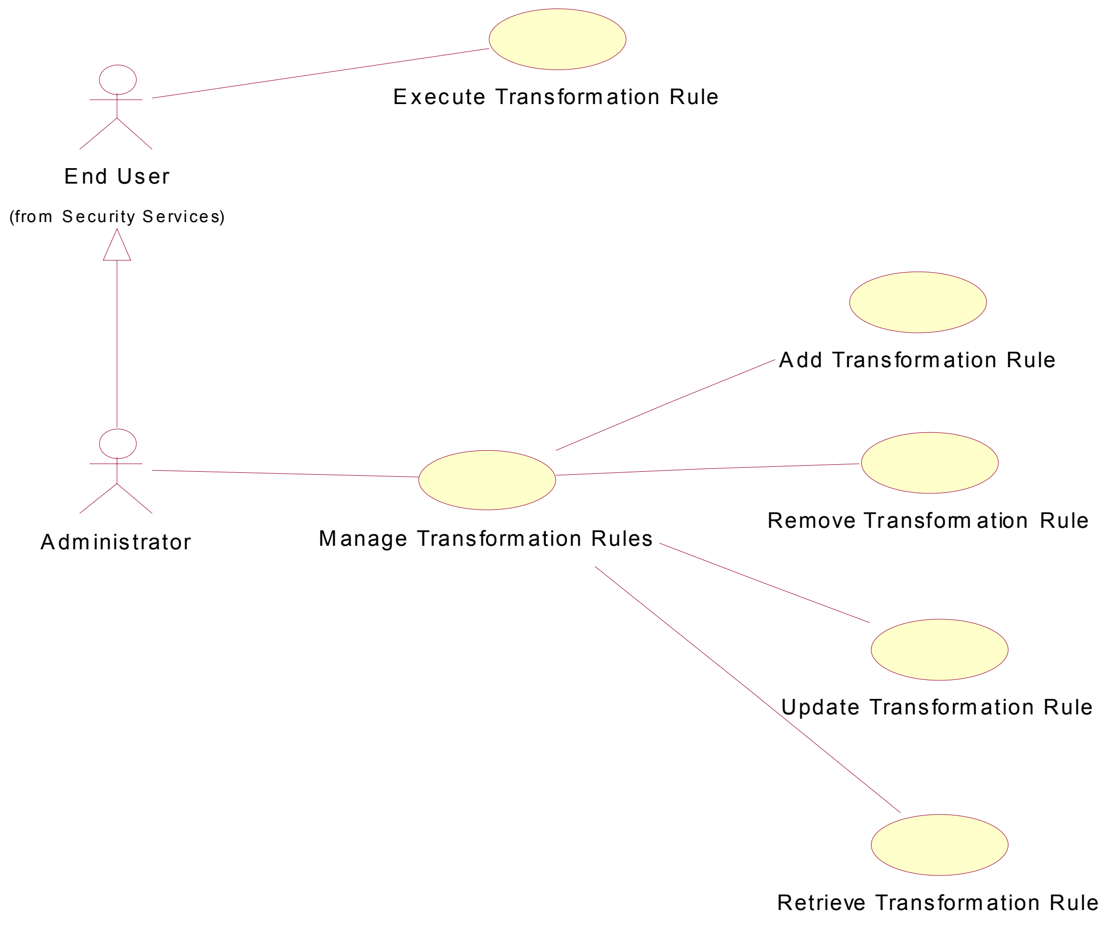
MessagingManagerException will map to a fault as specified in Section 4.3.6 of the JAX-RPC .8 specification.

Similar to the implementation patterns applied in other service components, two implementation specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. The MessagingManagerService is initially intended to implement the MessagingManager interface and serve as the entry point for incoming Web Service requests. The MessagingManagerService class will likely perform any necessary object assembly and forward the request to a central control where security, logging, and routing will occur. The BusinessManager interface will enable the MessagingManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller where services such as security and logging will occur.

3.2.5.5 Transformation Services

Transformation Services provide the functions required to enable bidirectional syntactic transformation of data between heterogeneous applications within different domains. These services reference the Information View of the TIA System Architecture and will identify the approach to semantic transformation between the domain objects.

3.2.5.5.1 *Transformation Services System Use Cases*

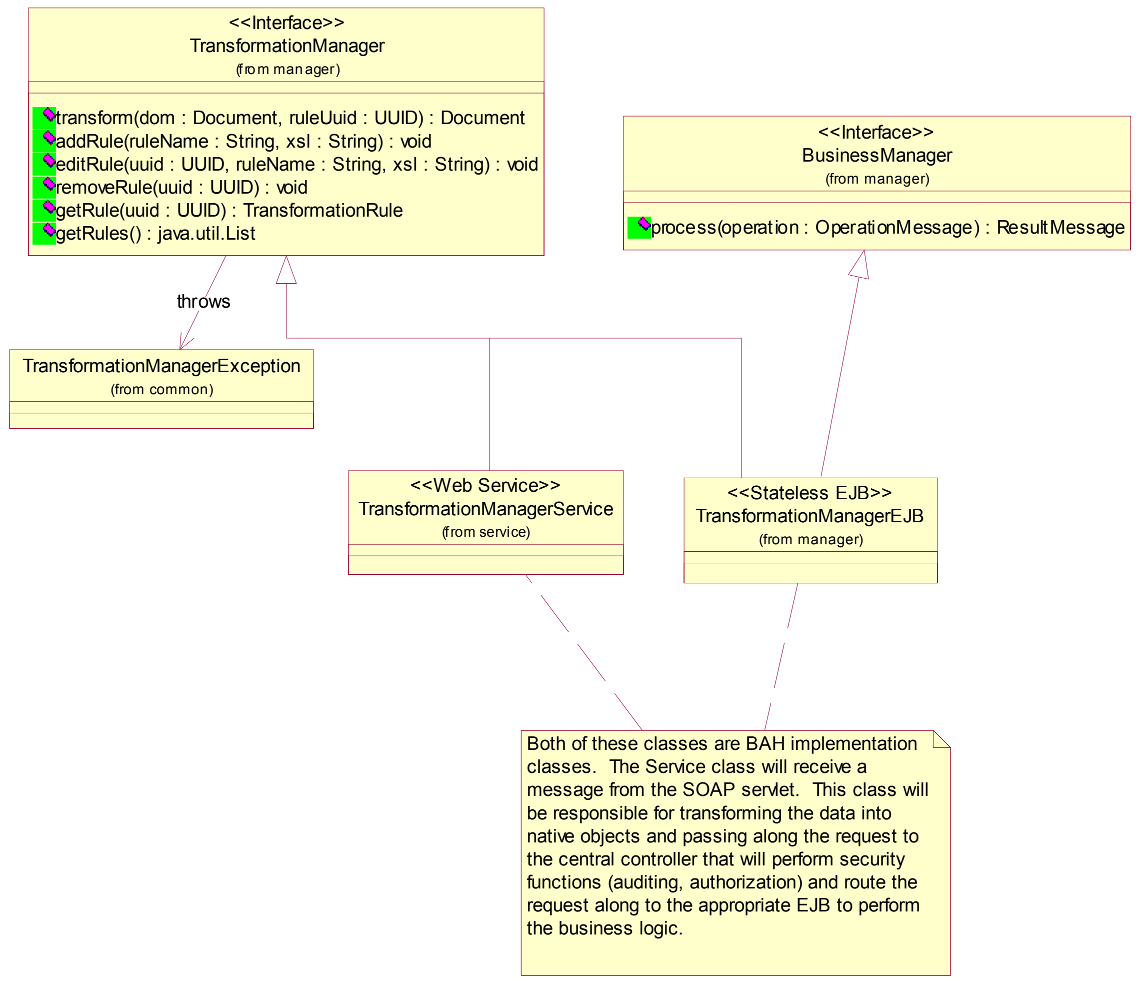

Execute Transformation Rule
End User
(from Security Services)
Add Transformation Rule
Remove Transformation Rule
Administrator
Manage Transformation Rules
Update Transformation Rule
Retrieve Transformation Rule

The Use Case diagram above highlights the initial set of functionality proposed for Transformation Services. The core set of functionality includes executing a transformation and managing the set of transformation rules.

Executing a transformation rule involves passing in an existing well-formed XML document and applying a transformation rule to the document to produce a new well-formed XML document. The transformation rules will utilize XSLT technology to perform the document translation. The management of transformation rules includes the ability to create, update, remove, and retrieve transformation rules from the system. The management of transformation rules can only be performed by authorized administrators of the system.

3.2.5.5.2 *Transformation Services Interface*

Based upon the Use Case diagram defined above, the following class diagram has been defined to satisfy the base requirements:


<<Interface>>
TransformationManager
(from manager)
transform(dom : Document, ruleUuid : UUID) : Document
addRule(ruleName : String, xsl : String) : void
editRule(uuid : UUID, ruleName : String, xsl : String) : void
removeRule(uuid : UUID) : void
getRule(uuid : UUID) : TransformationRule
getRules() : java.util.List
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
throws
TransformationManagerException
(from common)
<<Web Service>>
TransformationManagerService
(from service)
<<Stateless EJB>>
TransformationManagerEJB
(from manager)
Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.

The class diagram above depicts the initial high-level design representation of the TIA Transformation Services. An interface called "TransformationManager" has been created to serve as the service view to the external world. The TransformationManager interface exposes key methods to enable the transformation of an XML document and the basic administration of rules including CRUD functionality. The TransformationManager interface will be used to generate the WSDL service description that will be used by developers to develop programs that communicate with the service.

The TransformationManager is required to throw an exception called "TransformationManagerException" in the event a request cannot be fulfilled. The

TransformationManagerException will map to a fault as specified in Section 4.3.6 of the JAX-RPC .8 specification.

Similar to the implementation patterns applied in other service components, two implementation specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. The TransformationManagerService is initially intended to implement the TransformationManager interface and serve as the entry point for incoming Web Service requests. The TransformationManagerService class will likely perform any necessary object assembly and forward the request to a central control for security, logging, and routing. The TransformationManagerEJB class will likely be implemented as a stateless Enterprise Java Bean (EJB) and perform the necessary business logic to service the request. The diagram above reflects the TransformationManagerEJB implementing a BusinessManager interface. The BusinessManager interface will enable the TransformationManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller, where services such as security and logging will occur.

3.2.5.6 Computational Services

Many mission-critical TIA applications, such as those in artificial intelligence (AI), information mining (IM), and data warehousing (DW), are highly resource-intensive. Such applications require orchestrated access to a diverse collection of resources, including computational resources, storage resources, networks, programs, and databases. Traditionally, such applications often have to be hosted in a centralized, custom-designed environment with high-end server clusters or even supercomputers. Such an environment is generally homogeneous, static, non-interoperable, vendor-dependent, and costly to operate and maintain.

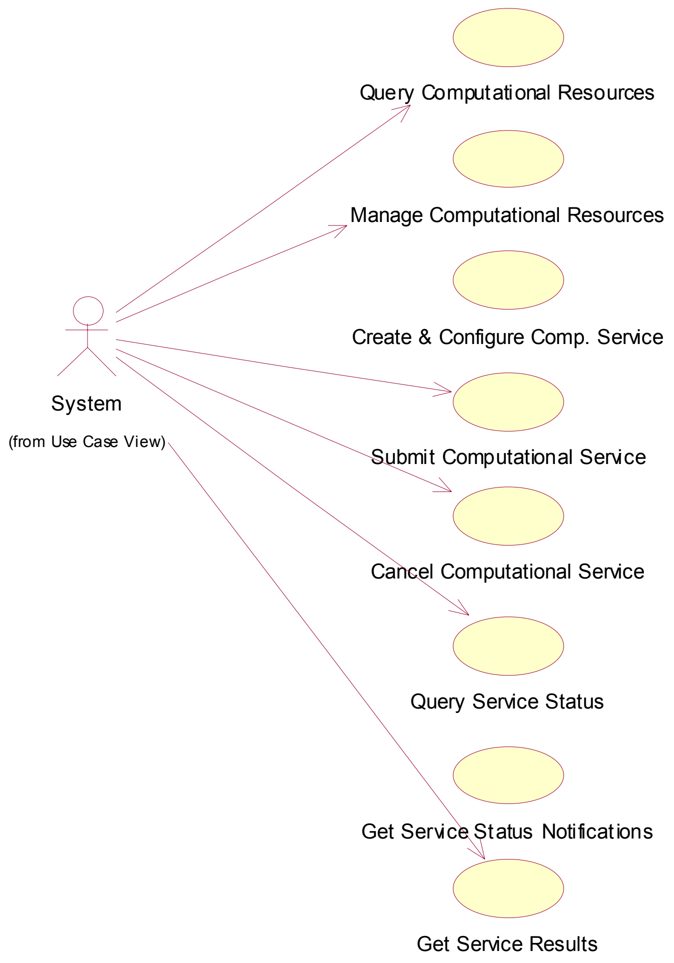
Computational Services uses recent technology advances in grid computing to give better support for such resource-intensive applications. It provides (1) a generic distributed architecture that allows for dynamic resource access across multiple heterogeneous platforms, and (2) a set of open, accessible interfaces to access and manage resource-intensive applications under this architecture.

Just like other TIA Core Services, Computational Services expresses its interfaces using Web Service protocols, which are programming language-, programming model-, and system software-neutral, to achieve maximum interoperability. Furthermore, Computational Services is built upon the OGSA model, the *de facto* standard for grid computing. Under the service-oriented OGSA architecture:

- Applications, resources, and so on are all represented as services.
- Services are dynamically created, discovered, and accessed using standard interfaces regardless of the native computing platform.
- Services are managed in a controlled, fault-resilient, and secure manner.

As a result, this architecture dramatically lowers the entry barrier to resource-intensive computing. Resources from heterogeneous networks, possibly across multiple administrative domains, can be organized into a massive "parallel computer," drastically improving application performance and response time.

3.2.5.6.1 Computational Services System Use Cases


Query Computational Resources
Manage Computational Resources
Create & Configure Comp. Service
Submit Computational Service
Cancel Computational Service
Query Service Status
Get Service Status Notifications
Get Service Results
System
(from Use Case View)

The Use Case diagram depicted above highlights the initial set of functionality proposed for computational services. The core set of functionality can be divided into two categories:

1. Use cases related to managing computational resources
2. Use cases related to managing computational services or jobs.

Computational resources are those used by a computational-intensive application, including but not limited to machines, CPUs, memory, storage, network bandwidth, and so on. Under the OGSA architecture, the relationship between resources and computational jobs are dynamically configured rather than statically assigned. The system provides a "resource directory" that keeps track of available computational resources and their properties. Querying this directory will provide the system with a set of resources that meet certain criteria. For example, it will answer the question such as "Which Red Hat 7.2 machines are available with a load of less than 30%?"

Management of computational resources involves creating, removing, and updating the status of different kinds of resources. All resources are described in structured XML, enabling users to construct queries using standard query languages such as XPath and XQuery. Depending on domain specific requirements, different implementations may be provided such as Relational Database (RDBMS) or Lightweight Directory Access Protocol (LDAP). Again, the Web Services APIs ensure that users always use the same interface to access the resource information regardless of the underlying implementation.

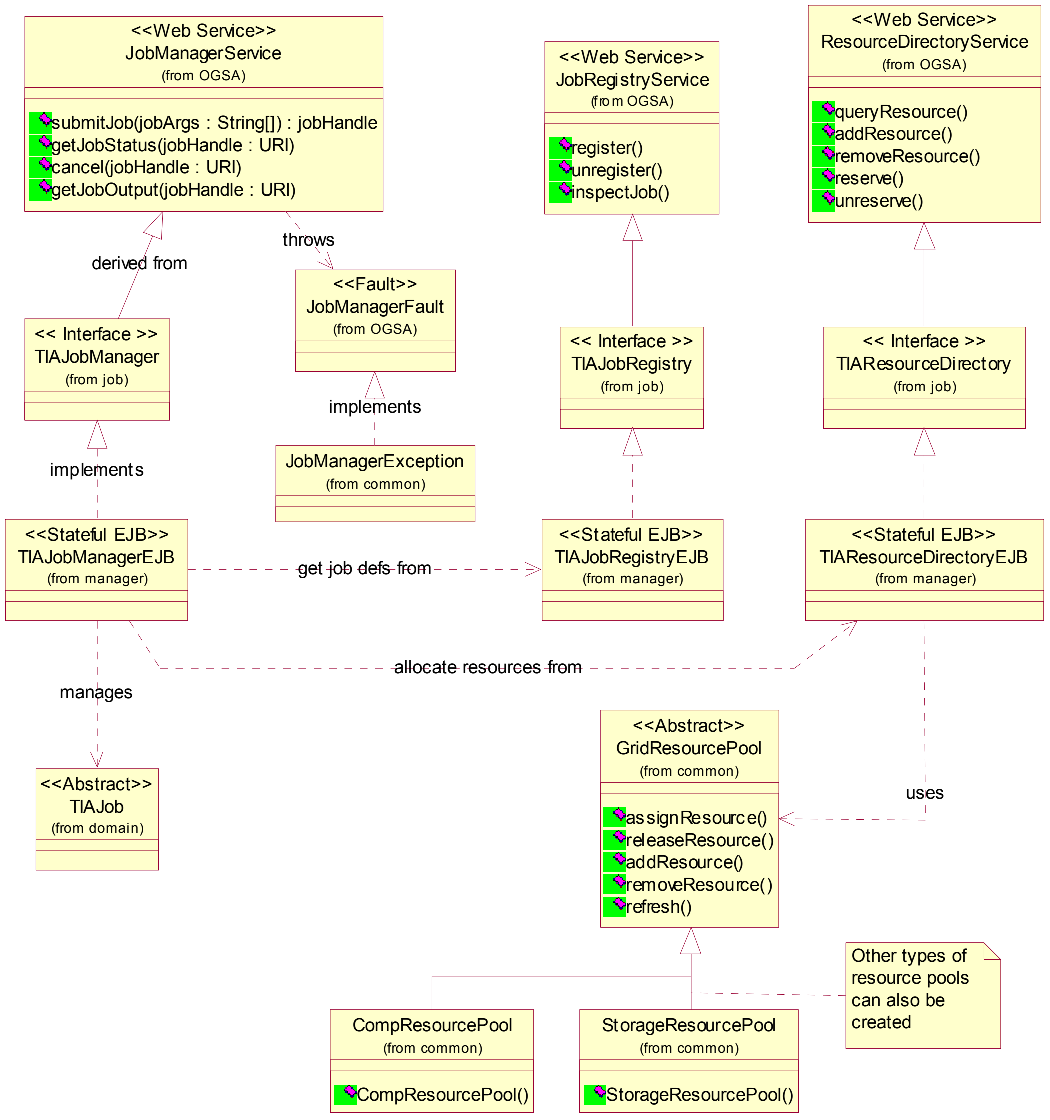
Before a computational job can be submitted and executed, the system needs to gather two kinds of information about this job, namely, (1) job configuration, such as the program(s) to run, environment variables, input and output requirements, and the like; and (2) resource requirements, such as machine type, OS level, memory, and so on. The "Create and Configure Computational Service" Use Case allows the system to gather such information.

The Submit Computational Service enables the system to submit a request to execute a computational job. Based on the job configuration and its resource requirements, a job manager will attempt to match and allocate resources from the resource directory, make preparations (e.g., I/O data) for the job, then submit the job for execution on the assigned resources. The OGSA architecture enables local or remote location transparency, alleviating users from the "plumbing" details such as network communication and data transport.

Likewise, the system provides other Use Cases for managing the lifecycle of a computational job, such as canceling a job, monitoring job status, and getting the job execution results.

It is worth noting that the Use Cases for job management are designed under an *asynchronous* model, whereby the external users do not have to wait for the service to be completed. Rather, clients may choose to subscribe to notifications or "callbacks" which is sent after the service is completed.

3.2.5.6.2 Computational Services Interface



The class diagram above represents the initial high-level design representation of the TIA Computational Services. The intention here is not to provide an exhaustive list of all implementation classes, but to identify interfaces that represent the key functionality described in the use case diagram and illustrate the design patterns that guide the implementation.

The JobManagerService is the key interface that clients will be dealing with, which provides operations for submitting a job, canceling a job, inquiring job status, and getting job outputs.

The JobManagerService uses JobRegistryService to manage the definitions of computation-intensive jobs, which include job execution environment properties, I/O requirements, resource requirements, and various preferences.

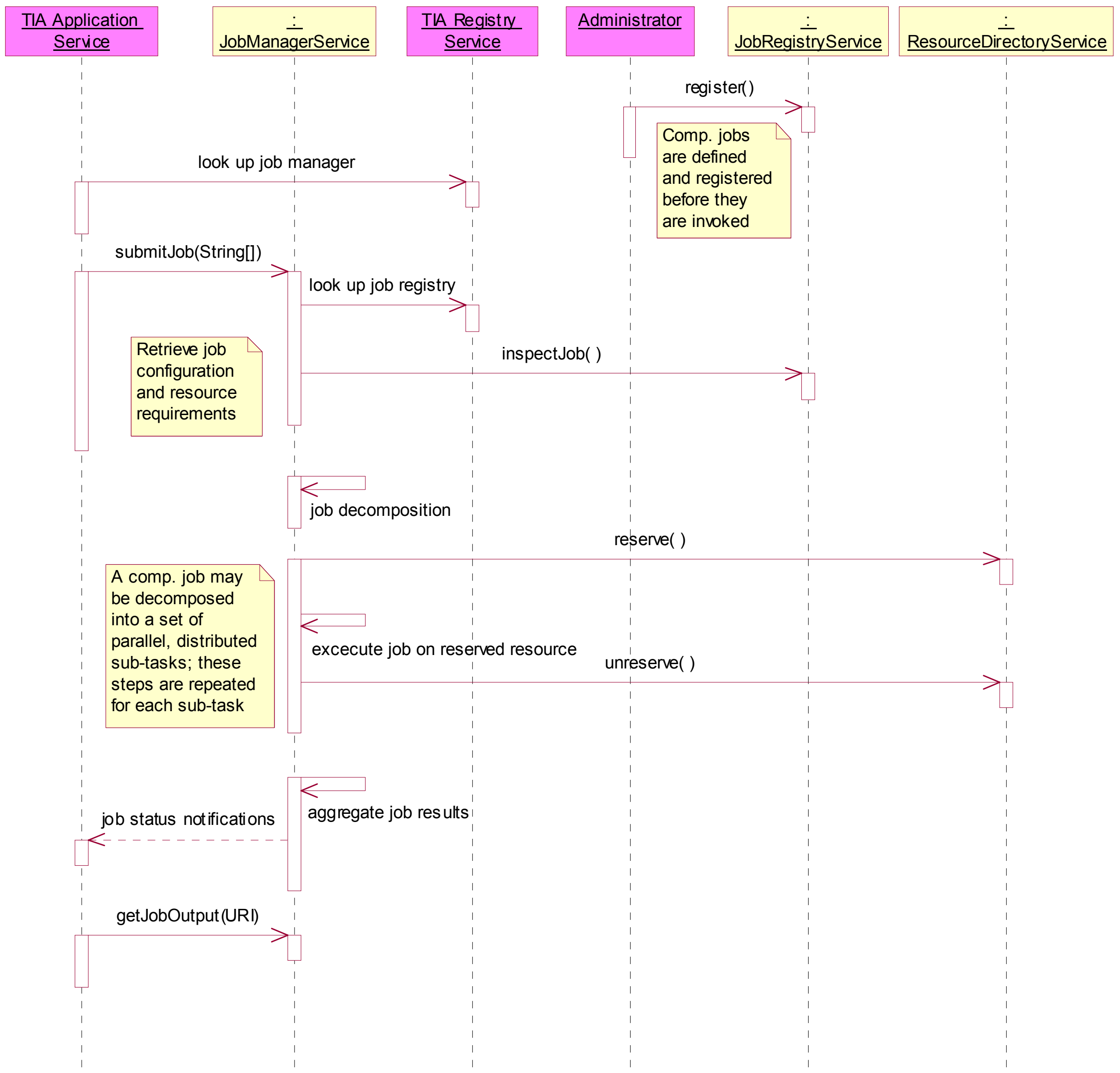
A ResourceDirectoryService keeps track of all available resources in the computational "grid." In addition to storing resource details (e.g. CPU, memory, and storage info for a computer), it also receives dynamic updates to reflect resource availability.

The class diagram is self-explanatory to a large extent, but the following points are worth noting:

- Web Services interface definitions, as defined in the WSDL language in the next section, represent the official "contract" between clients and service providers. Java interfaces and EJB implementations are derived from the WSDL interfaces.
- Likewise, Web Services "faults" represent abnormal or error conditions encountered in the system and are thrown back to the invoker of a service. These faults map naturally to Java exceptions.
- Different implementations can be provided for job management and resource management, depending on the nature of the applications of interest and domain-specific requirements. For example, as illustrated in the class diagram, the Java implementation can use different resource pools internally to manage computational resources, storage resource, etc. Alternatively, the TIAResourceRegistryEJB can serve merely as a delegation point and hand off requests to external resource managers (e.g., in LDAP). Regardless of the implementation, the Web Services interface remains the same—a key benefit of the SOA.

3.2.5.6.3 Computational Services Sequence Diagrams

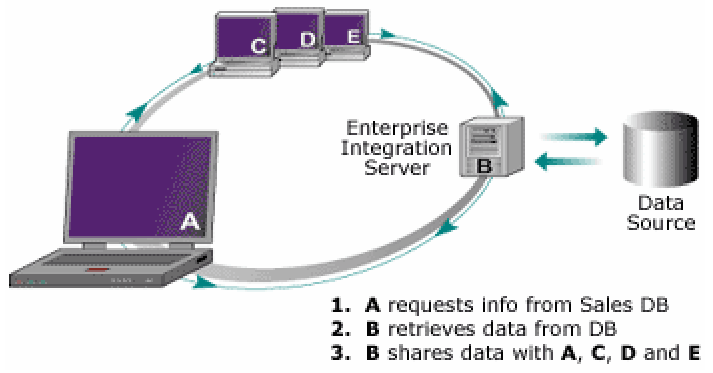
The following sequence diagram illustrates how an upper-level Application Service might use the Computational Services. It shows how the components listed in the Class Diagram above interact with one another:


TIA Application Service
: JobManagerService
TIA Registry Service
Administrator
: JobRegistryService
: ResourceDirectoryService
register( )
Comp. jobs are defined and registered before they are invoked
look up job manager
submitJob(String[])
look up job registry
Retrieve job configuration and resource requirements
inspectJob( )
job decomposition
reserve( )
A comp. job may be decomposed into a set of parallel, distributed sub-tasks; these steps are repeated for each sub-task
excecute job on reserved resource
unreserve( )
aggregate job results
job status notifications
getJobOutput(URI)

3.2.5.7 Edge Gateway Services

Edge Gateway Services are designed to bridge the gap between traditional center and edge-based computing. Edge Gateway Services will likely be implemented and deployed using commercially available software products from Groove Networks.

Today, Groove Networks offers a product offering, Enterprise Integration Server (EIS), designed to provide edge-based applications with the ability to securely share data with a variety of external systems including knowledge management systems, document management repositories, relational databases, and other legacy applications. The following diagram extracted from http://www.groove.net/products/enterprise/bot/ highlights the basic workflow of integrating external resources with a Groove shared space using the EIS.


C
D
E
Enterprise Integration Server
B
A
Data Source
1. A requests info from Sales DB
2. B retrieves data from DB
3. B shares data with A, C, D and E

In the diagram above, EIS "bot" technology is employed to broker the transfer of data between the shared space and the enterprise data source. Here, a "bot" is invited to a shared space similar to any other user on the network, however, the bot's sole responsibility is to perform a task such as retrieving data from a center-based relational database and placing the data into the shared space. An architecture employing Edge Gateway Services will likely have multiple bot services deployed including services for backup/archiving, logging, and search indexing.

Moving forward, Groove Networks is planning a new product offering currently coined Edge Services. Edge Services will introduce a SOAP-based "relay" server into the architecture. This mechanism will extend the participation in a shared space to users outside of the traditional Groove transceiver. Portals and portable device users will now have access to data stored in a Groove shared space.

3.2.6. Information View

The Information View describes the data that traverses the system. This section is broken into the Semantic View, which is the understanding of the data (i.e., what the data means), and the Syntactic View (i.e., how the data is physically represented across the system). Colloquially, the Syntactic View models the file formats, protocols, and interface definitions that exist within the architecture. The Semantic View captures the mechanisms of meaning exchange within the system.

3.2.6.1 Semantic View

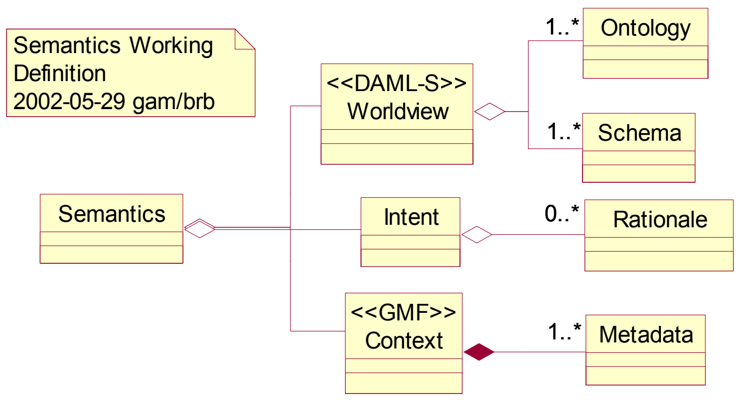
The Semantic View represents the *semantic architecture* of the TIA System. It is comprised of a means for accessing content semantics through services, and a mechanism for communicating semantics along with the content and services. Initially, this view contains a model of business rule types within the system and the Genoa Metadata

Framework (GMF). It will be expanded to contain DAML-S descriptions, RDF models of context, and other semantic models.

A semantic interface in this architecture will contain a description of semantics, including a Context representation (see Section 3.2.6.2.1), intent representation (rationale), and a Worldview representation (DAML-S), accessible through a service description. Operations include the ability to synchronize context with external sources. Context is represented using the Genoa Metadata Framework (GMF). Intent can be captured using an RDF extension to the Context for the text rationale.

3.2.6.1.1 Semantics Working Definition

Efforts such as the Semantic Web is based on Berners-Lee's notion that "semantic" means "machine-processable". [1] The object is to provide enough "meaning" bundled with the delivery of the data that applications will "do the right thing" with it. An example is providing a semantic description of a travel reservations service enabling a software agent to make airline bookings on behalf of a human actor. Service Registries are built on the process of exposing descriptions of the service to inquiring applications. However, current approaches (e.g. UDDI[2], ebXML Registry[3]) are very limited semantically, revealing little more than short text descriptions, flat attributes (e.g., JINI), and interface syntax. A smaller (but increasing rapidly) number of approaches attempt to provide higher-level semantic service matching references. However, in end-to-end systems involving extensive interaction of Actors in complex decision-making or analyses, "doing the right thing" requires an even richer representation of the semantics of the content.



Semantics Working Definition

[1] T,Berners-Lee. Keynote on the Semantic Web at XML 2000, Dec 2000. (http://www.w3.org/2000/Talks/1206-xml2k-tbl)

[2] UDDI Technical White Paper. Sep. 2000. (http://www.uddi.org/)

[3] ebXML Project Team. "Using UDDI to Find ebXML Reg/Reps". White Paper. Sep. 2001. (http://www.ebxml.org/)

The Semantic View uses a definition of semantics that closer to the natural language sense of "meaning" than Berners-Lee's. The previous figure illustrates this definition of semantics.

First, consider the *Worldview* upon which the content is based. As previously mentioned, there is significant current work in communicating ontologies and/or schemas (e.g. DAML+OIL and others). Ontologies support the definitions necessary to create the vocabularies and units of measurement. This approach builds on this work.

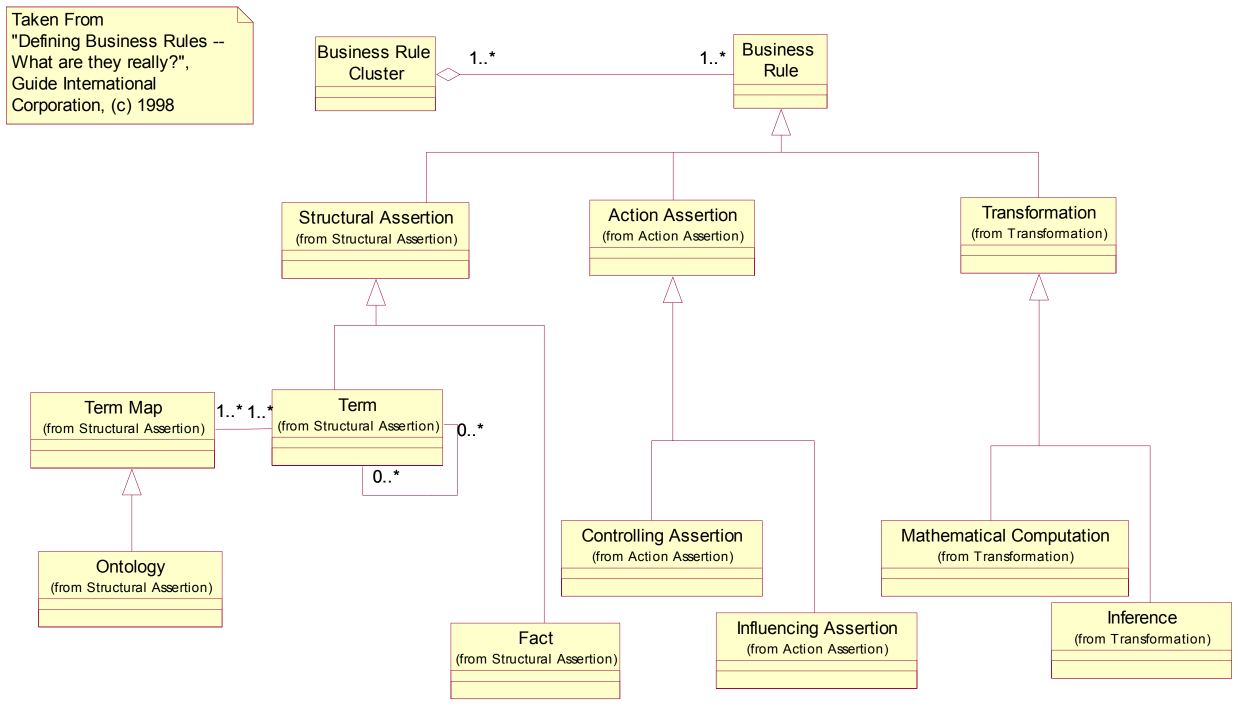
The second part of the definition of semantics in this paper that is important in complex situations is the *Intent*. This area is beginning to gather significant interest, especially within the military. Intent answers the question "Why?". It is the reasoning behind the content and is an area needing significant research.

The last part of the definition, *Context*, is the aggregate of the metadata associated with some specific content. It is an aspect of meaning that is rarely included in semantic models. Including Context enables the definition to take into account the environment of content and the services operating on it. Context is what makes content information instead of just data. Context can answer the questions "Where?, How?, When?, and Who?". Indexing on context allows for the constructive accumulation of content that provides opportunities create information value greater than the sum of the constituent content.

Finally, Content bundled with its context has intrinsic value. The value of content to a particular user is determined by its relevance and accessibility. Context can be used to establish both relevance (e.g., version, source, annotations) and accessibility (e.g., mime type, url). Therefore, the package of content and its context intrinsically contains the mechanisms for the user to assess the value of the content.

3.2.6.1.2 Business Rules

The Business Rules package of the Business Information View captures the types of business rules related to the business processes and information. This decomposition of rules will be used to represent the business rules within the services defined in the Business Application Services View. The structure of business rules is shown below.


Taken From "Defining Business Rules -- What are they really?", Guide International Corporation, (c) 1998
Business Rule Cluster
1..*
1..*
Business Rule
Structural Assertion (from Structural Assertion)
Action Assertion (from Action Assertion)
Transformation (from Transformation)
Term Map (from Structural Assertion)
1..*
1..*
Term (from Structural Assertion)
0..*
0..*
Ontology (from Structural Assertion)
Fact (from Structural Assertion)
Controlling Assertion (from Action Assertion)
Influencing Assertion (from Action Assertion)
Mathematical Computation (from Transformation)
Inference (from Transformation)

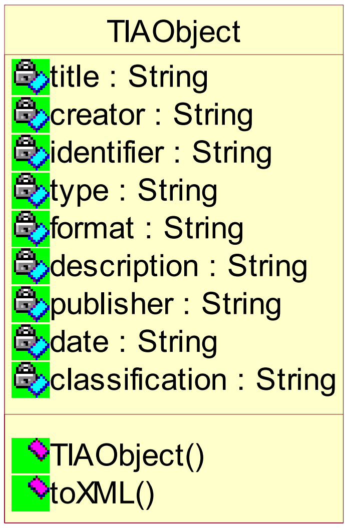
Business Rules Structure

3.2.6.2 Syntactic View

The Syntactic View of the information model describes the data as it is physically represented within the TIA system. This representation is necessary as it provides the basis for a unified security model and the ability to perform transformation services between data in disparate domains or applications. This section is organized into the description of the base object within the TIA domain that contains the common metadata attributes to all domain objects. Further sections will be added describing information specific to each domain and/or application within the TIA system.

3.2.6.2.1 TIA Domain Root Object

Within the TIA domain, all information that passes between systems via services, which will have a core set of common information. This information, or metadata, is fundamental to a common security and privilege model and to interoperability between services and domains. Based upon the TIA Metadata Framework described in the Semantic View, a base domain object, the TIAObject, has been defined and is shown



below.

The TIAObject includes the ability to serialize and deserialize an instantiation into XML as required to exchange information with and be operated upon Application Services. The XML Schema equivalent for TIAObject is shown in Section 3.2.6.2.1.1.

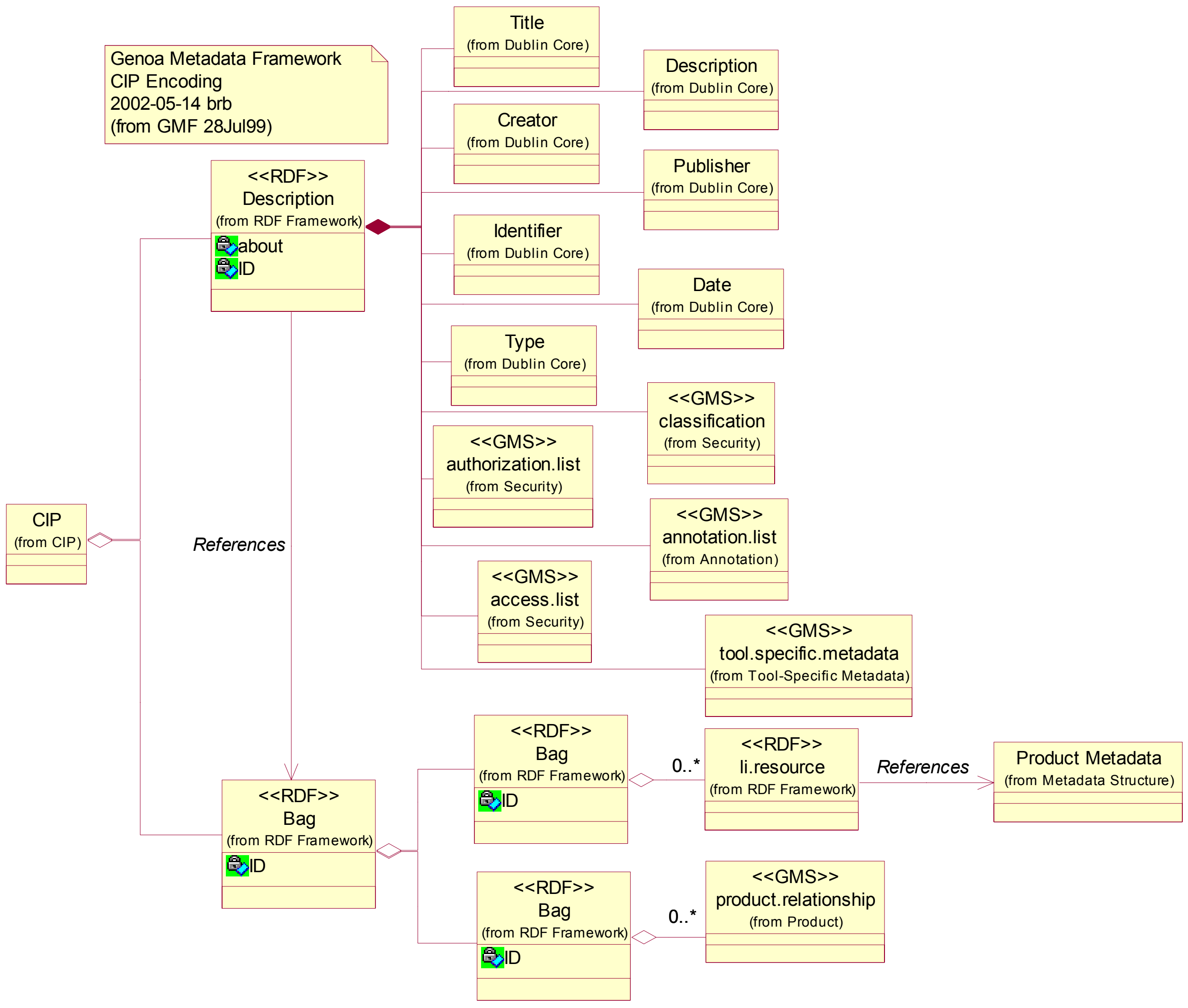
3.2.6.2.1.1 TIA Object XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="TiaObject">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="title"/>
                <xs:element ref="creator"/>
                <xs:element ref="identifier"/>
                <xs:element ref="type"/>
                <xs:element ref="format"/>
                <xs:element ref="description"/>
                <xs:element ref="publisher"/>
                <xs:element ref="date"/>
                <xs:element ref="classification"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="classification" type="xs:string"/>
    <xs:element name="creator" type="xs:string"/>
    <xs:element name="date" type="xs:string"/>
    <xs:element name="description" type="xs:string"/>
    <xs:element name="format" type="xs:string"/>
    <xs:element name="identifier" type="xs:string"/>
    <xs:element name="publisher" type="xs:string"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="type" type="xs:string"/>
</xs:schema>
```

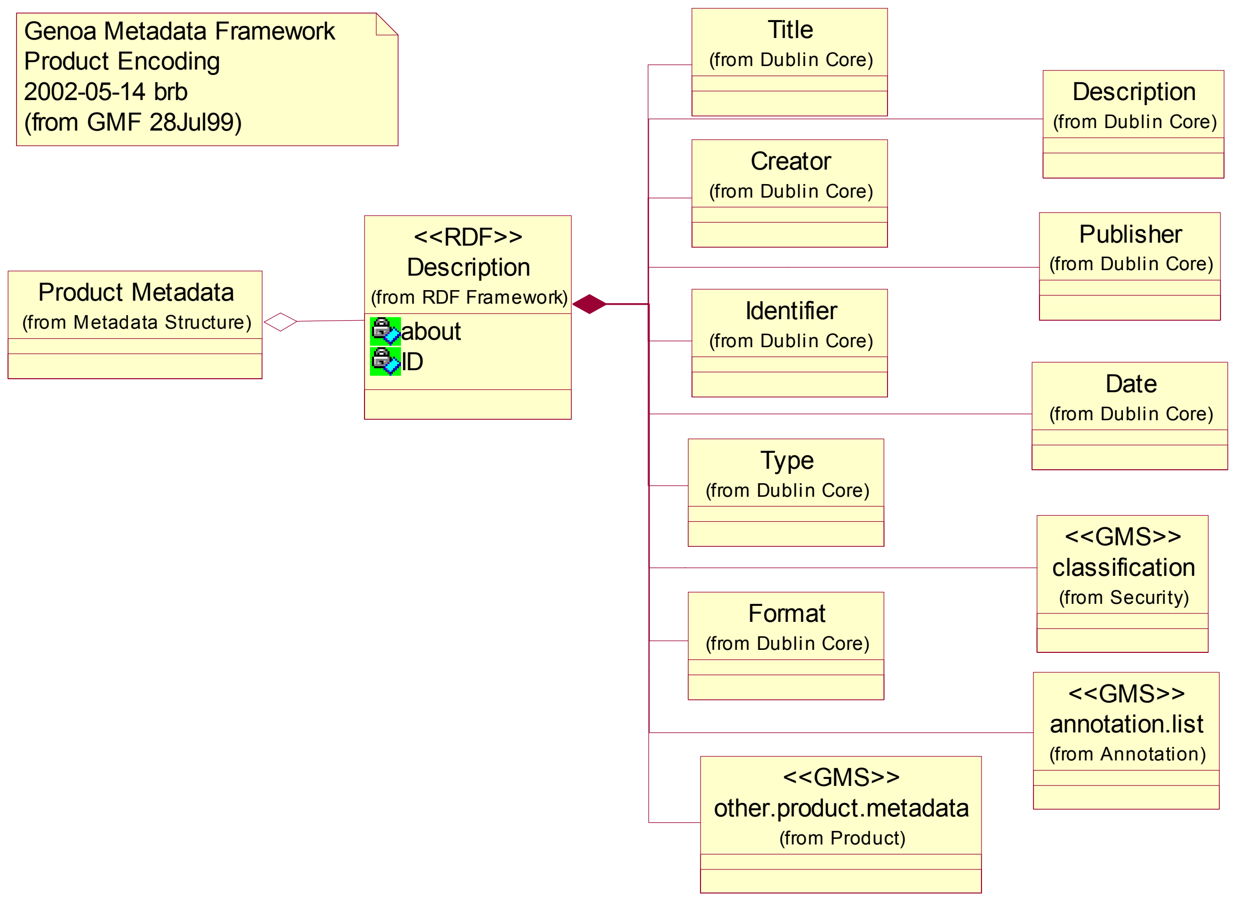
3.2.6.2.2 Genoa Metadata Framework

The Genoa Metadata Framework is uses the Resource Description Framework (RDF) to define an XML schema for representing metadata on individual files, products, and collections of products, creating Critical Information Packages (CIP). The diagrams shown in the following sections represent a UML representation of an XML Schema.

3.2.6.2.2.1 Critical Information Package


Genoa Metadata Framework
CIP Encoding
2002-05-14 brb
(from GMF 28Jul99)
<<RDF>> Description (from RDF Framework)
about
ID
Title (from Dublin Core)
Description (from Dublin Core)
Creator (from Dublin Core)
Publisher (from Dublin Core)
Identifier (from Dublin Core)
Date (from Dublin Core)
Type (from Dublin Core)
<<GMS>> classification (from Security)
<<GMS>> authorization.list (from Security)
<<GMS>> annotation.list (from Annotation)
<<GMS>> access.list (from Security)
<<GMS>> tool.specific.metadata (from Tool-Specific Metadata)
CIP (from CIP)
References
<<RDF>> Bag (from RDF Framework)
ID
<<RDF>> Bag (from RDF Framework)
ID
0..*
<<RDF>> li.resource (from RDF Framework)
References
Product Metadata (from Metadata Structure)
<<RDF>> Bag (from RDF Framework)
ID
0..*
<<GMS>> product.relationship (from Product)

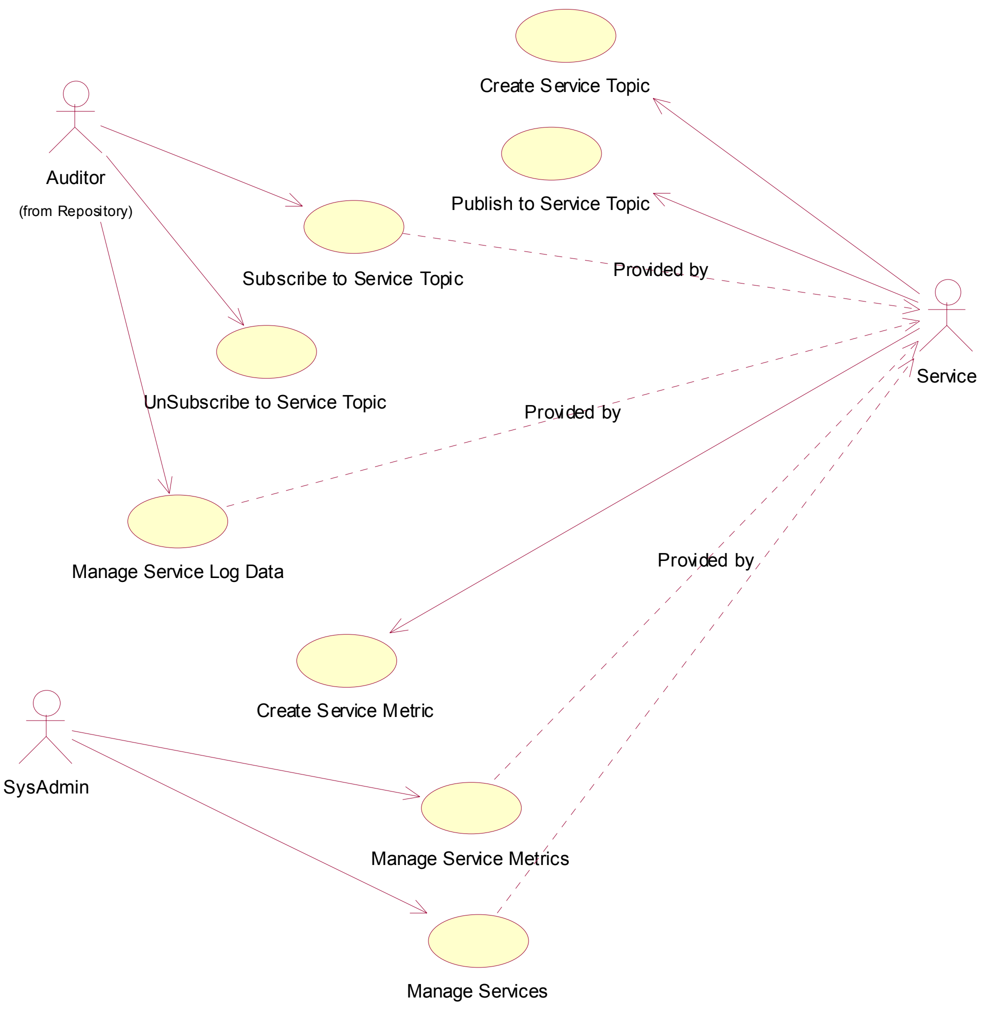
3.2.6.2.2.2 Product Metadata



3.2.7. Services Management

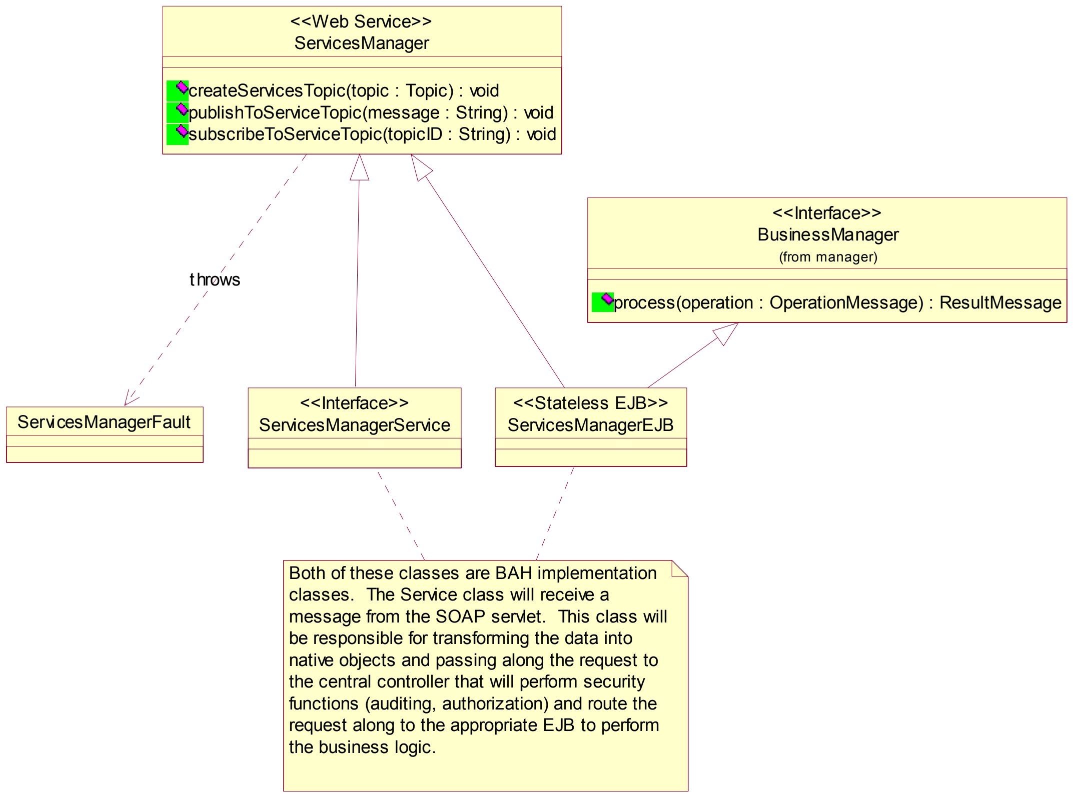
Services Management is a critical component to the SOA. This functional area will allow a centralized approach to manage information on the services being utilized within the TIA system. This information can be utilized to identify the performance of TIA services to allow the services administrator the ability to normalize the system by possible distributing the load across multiple servers. Additionally, this information can be used to audit analyst access to information to help with data privacy issues, should they arise. A tertiary category of services management is in the context of pattern analysis and notification. In this category, analysts' usage patterns can be detected and thus act as an additional intelligence source notifying, for example, that there is a high degree of activity within a specific service topic. The latter category of services management will be reserved for future expansion.

3.2.7.1 Services Management System Use Cases



This diagram depicts the functions that we view would be required to perform Services Management. The actors portrayed here show the different categories of users of services management. For example, the system administrator would be particularly interested in the performance metrics of the service invocations. Sub-level Use Cases such as creating, updating, and removing metrics will be incorporated. We envision metrics such as number of service requests per minute, service latencies, and service downtime could be extracted. In addition, the system administrator would need to manage the individual services such as configuration, deployment, removal, and so on. It is envisioned that a services management console would be developed to provide the ability to perform these service management functions. Furthermore, auditors could gain access into privacy act issues and view the usage of personalized data, if it becomes available. In future documentation, we will provide the software architecture of the services management console that is monitoring the Core and Application Services.

3.2.7.2 Services Management Interface


<<Web Service>>
ServicesManager
createServicesTopic(topic : Topic) : void
publishToServiceTopic(message : String) : void
subscribeToServiceTopic(topicID : String) : void
throws
<<Interface>>
BusinessManager
(from manager)
process(operation : OperationMessage) : ResultMessage
ServicesManagerFault
<<Interface>>
ServicesManagerService
<<Stateless EJB>>
ServicesManagerEJB
Both of these classes are BAH implementation classes. The Service class will receive a message from the SOAP servlet. This class will be responsible for transforming the data into native objects and passing along the request to the central controller that will perform security functions (auditing, authorization) and route the request along to the appropriate EJB to perform the business logic.

The class diagram above represents the initial high-level design representation of the TIA Services Management interface. The exposed service only contains centralized interface functions that the Core and Application Services will access to provide their service statistics. This is based on a messaging-type approach rather than the services manager having to poll all the services. The detail of the services management application logic and user interface will be provided in a supplemental application architecture document.

We have created an interface called "ServicesManager" to serve as the Service View to the external world. The ServicesManager interface exposes key methods required to create and publish to a service topic. This is the initial set of external services available to support auditing and performance analysis. As services management evolves, additional interface functions will be provided. The ServicesManager interface will be used to generate the WSDL service description that will be used by developers to develop programs that communicate with the service.

The ServicesManager is required to throw a fault called "ServicesManagerFault" in the event a request cannot be fulfilled. The ServicesManagerFault will map to an exception as specified in Section 4.3.6 of the JAX-RPC .8 specification[1].

Similar to the implementation patterns applied in the other service components, two implementation specific classes have been included in the diagram to serve as placeholders until the detailed design details have been determined. The initial intent of the ServicesManagerService is to implement the ServicesManager interface and serve as the entry point for incoming web service requests. The ServicesManagerService class will likely perform any necessary object assembly and forward the request to a central control where security, logging, and routing will occur. The ServicesManagerEJB class will likely be implemented as a stateless Enterprise Java Bean (EJB) and perform the necessary business logic to service the request. The diagram above reflects the ServicesManagerEJB implementing a BusinessManager interface. The BusinessManager interface will enable the ServicesManagerEJB to be plugged seamlessly into the TIA framework and integrate with the central controller where services such as security and logging will occur.

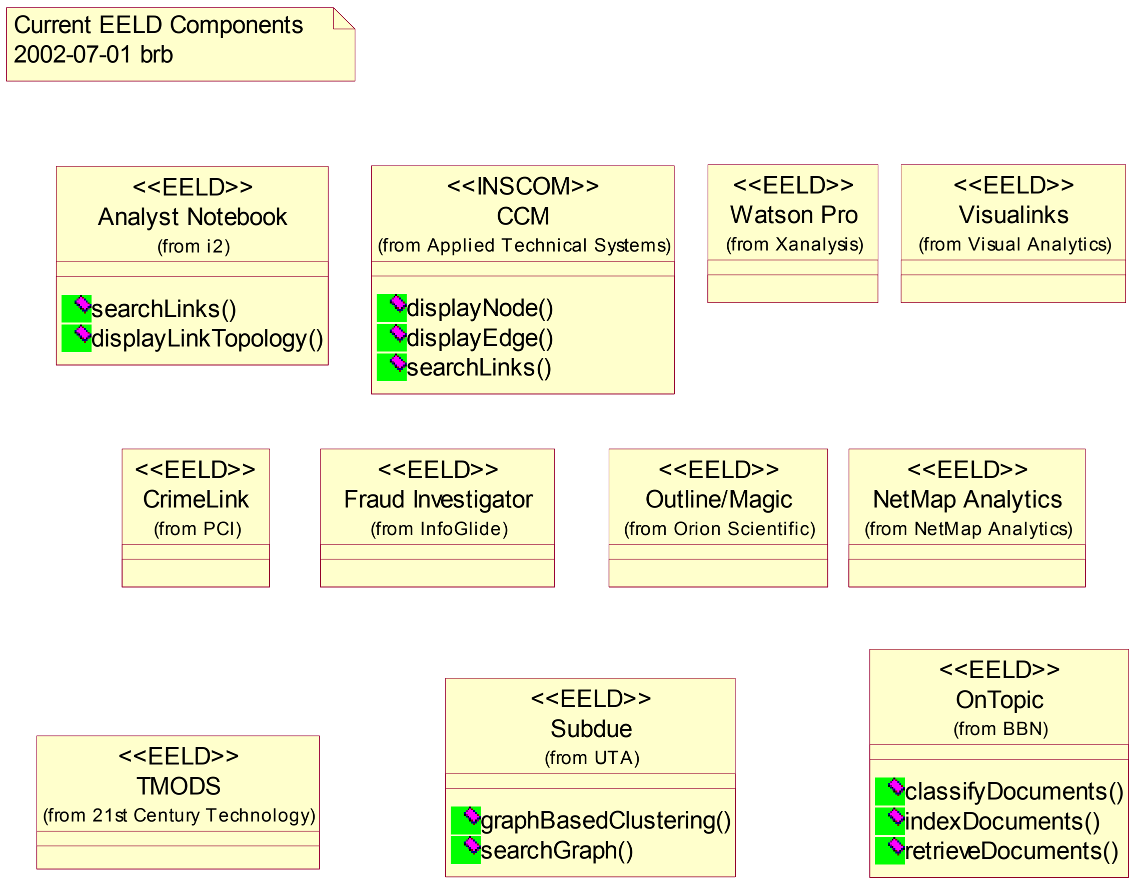
3.2.8. Component Catalog View

The Components Class View is a catalog containing the static definitions of commercial technology components and those produced by DARPA programs. It is arranged in packages listed by provider.

3.2.8.1 Evidence Extraction and Link Discovery (EELD)

The goal of the EELD program is development of technologies and tools for automated discovery, extraction and linking of sparse evidence contained in large amounts of classified and unclassified data sources. EELD is developing detection capabilities to extract relevant data and relationships about people, organizations, and activities from message traffic and open source data. It will link items relating potential terrorist groups or scenarios, and learn patterns of different groups or scenarios to identify new organizations or emerging threats.[4]

[4] More information about the EELD Program is available at http://www.darpa.mil/iao/EELD.htm.



3.2.8.1.1 Analyst Notebook

Analyst Notebook is visual investigative analysis software.[5]

3.2.8.1.2 CCM

CCM software provides intelligent search, visual navigation, automatic correlation.[6]

3.2.8.1.3 CrimeLink

Crimelink software includes matrix manipulation, link charts, time event charting, and telephone toll analysis.[7]

3.2.8.1.4 Fraud Investigator

[5] More information about Analyst Notebook is available at http://www.i2.com/.

[6] More information about CCM is available at http://www.apptechsys.com/.

[7] More information about CrimeLink is available at http://www.crimelink.com/.

Fraud Investigator performs similarity searches to find data and similarities in data that other search technologies are unable to uncover using the Similarity Search Engine (SSE).[8]

3.2.8.1.5 NetMap Analytics

NetMap Analytics finds relevant links in large amounts of data.[9]

3.2.8.1.6 Outline/Magic

Outline/Magic produces indexed documents, highlighted text, and link diagrams of concepts.[10]

3.2.8.1.7 OnTopic

OnTopic, developed under the EELD program, accepts either text input or speech that has been converted to text by our speech recognition system. The topic classification process can produce multiple topics from a list of up to 5,000 topics.[11]

3.2.8.1.8 TMODS

TMODS is being developed under the EELD program by 21st Century Technologies.

3.2.8.1.9 VisualLinks

VisualLinks uncovers the interactions and relationships between terrorist groups and its members.[12]

3.2.8.1.10 Watson Pro

Watson Pro is a link analysis product with embedded database connectivity. Also works with Xanalysis' entity extraction tool Quenza.[13]

3.2.8.2 Genoa (I)

Project Genoa is developing tools and a system for collaborative crisis understanding and management for the national security community including Commanders of the Unified Commands.[14]

[8] More information about Fraud Investigator is available at http://www.infoglide.com/.

[9] More information about NetMap Analytics is available at http://www.netmapsolutions.com/.

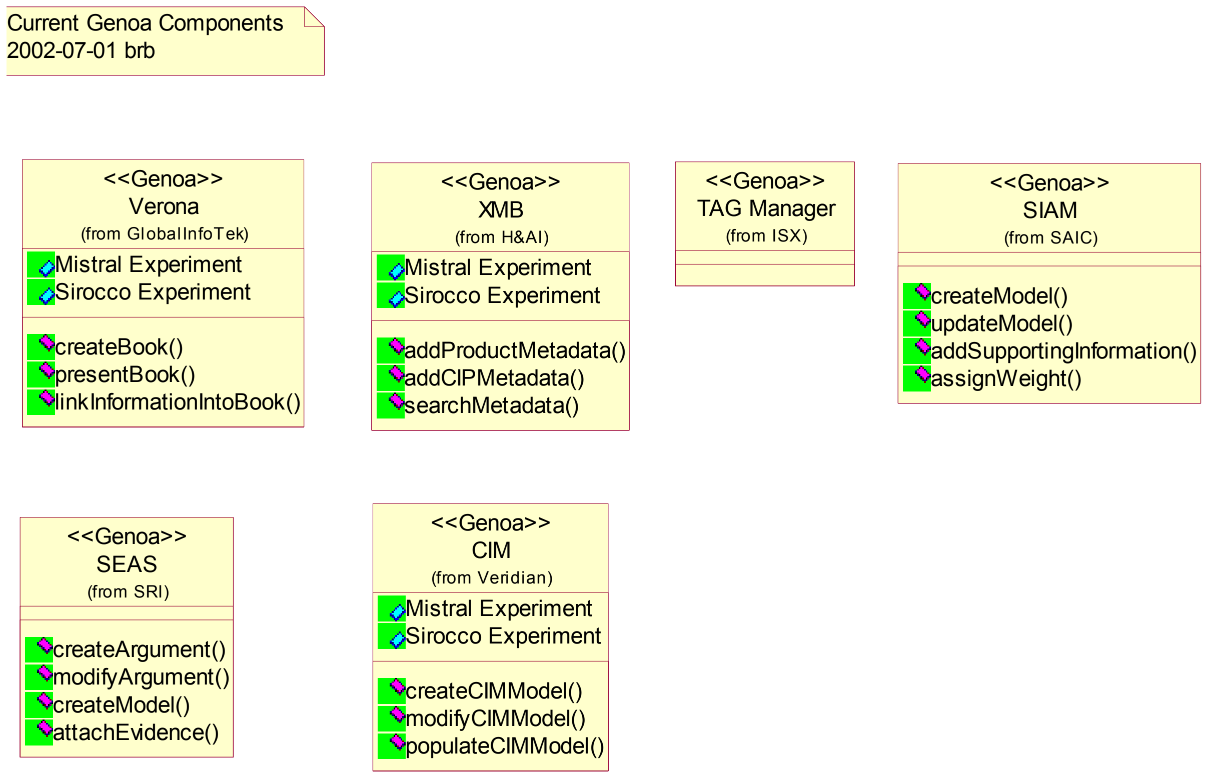
[10] More information about Outline/Magic is available at http://www.orionsci.com/.

[11] More information about OnTopic is available at http://www.bbn.com/speech/ontopic.html.

[12] More information about VisualLinks is available at http://www.visualanalytics.com/.

[13] More information about WatsonPro is available at http://www.xanalys.com/watson.html.

[14] More information about the Genoa Program is available at http://www.darpa.mil/iao/Genoa.htm.



3.2.8.2.1 Critical Intent Modeler (CIM)

CIM was initially developed under Project Genoa (I) by the Veridian Corporation.

3.2.8.2.2 Structured Evidential Argumentation System or SRI Early Alert System (SEAS)

SEAS is designed to aid intelligence analysts in predicting potential opportunities/crises. It is implemented as a web server that supports the construction and exploitation of a corporate memory filled with analytic products, methods, and their interrelationships, indexed by the situations to which they apply.[15]

3.2.8.2.3 Situation Influence Assessment Module (SIAM)

SIAM is a software application designed to assist people in analyzing complex problems and issues, especially when empirical information is sparse or uncertain. SIAM can be used in a range of operational situations, from corporate decision making to national security planning.[16]

3.2.8.2.4 Thematic Argument Group (TAG) Manager

[15] More information about SEAS is available at http://www.ai.sri.com/~genoa/help/about.html.

[16] More information about SIAM is available at http://www.inet.saic.com/inet-public/siam.htm.

The TAG Manager is an application developed under Project Genoa (I) by the ISX Corporation.

3.2.8.2.5 *Verona*

Verona improves the way users create, organize, share and distribute knowledge. Users can quickly access diverse information from multiple distributed sources. Users can easily author and deliver the information in a context tailored to their needs.

Verona makes use of the familiar notebook metaphor, appearing on users' desktops as a spiral binder separated into different tabbed chapters. The software allows users to easily customize and categorize their information.[17]

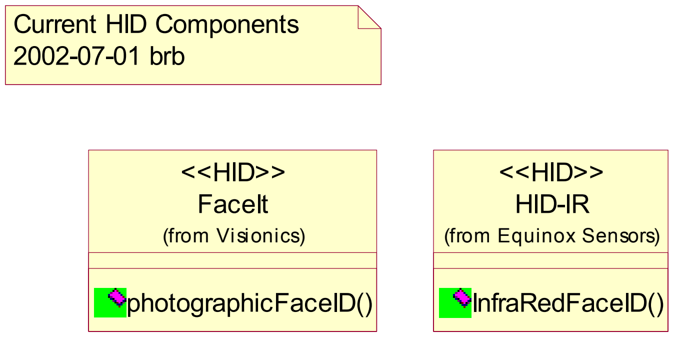
3.2.8.2.6 *XMB*

The XMB (XML Metadata Browser) was developed under Project Genoa (I) by Syntek Corporation. It is currently maintained by Hicks and Associates, Inc.

[17] More information about Verona is available at http://www.gitisolutions.com/.

3.2.8.3 Human Identification at a Distance (HID)

The goal of the HID program is to develop automated biometric identification technologies to detect, recognize, and identify humans at great distances. These technologies will provide critical early warning support for force protection and homeland defense against terrorist, criminal, and other human-based threats, and will prevent or decrease the success rate of such attacks against DoD operational facilities and installations. Methods for fusing biometric technologies into advanced human identification systems will be developed to enable faster, more accurate and unconstrained identification of humans at significant standoff distances.[18]



3.2.8.3.1 FaceIt

FaceIt is a facial recognition software engine that allows computers to rapidly and accurately detect and recognize faces.[19]

3.2.8.3.2 Human Identification at a Distance-Infrared (HID-IR)

HID-IR uses thermal imaging sensors to develop technologies to help improve sensitivity and resolution. Our sensors offer exciting new opportunities for biometric identification.[20]

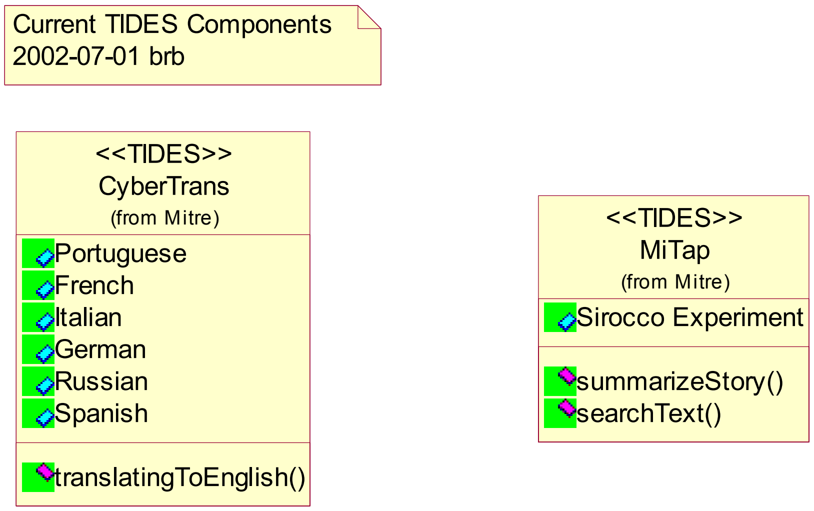
[18] More information about the HID Program is available at http://www.darpa.mil/iao/HID.htm.

[19] More information about FaceIt is available at http://www.identix.com/products/pro_sdks_faceit_what.html.

[20] More information about HD-IR is available at http://www.equinoxsensors.com/products/HID.html.

3.2.8.4 Translingual Information Detection, Extraction, and Summarization (TIDES)

The Translingual Information Detection, Extraction, and Summarization (TIDES) program is developing advanced language processing technology to enable English speakers to find and interpret critical information in multiple languages without requiring knowledge of those languages.[21]



3.2.8.4.1 MITRE Text and Audio Processing (MiTap)

The MiTAP system supports shared situational awareness through collaboration, allowing users to submit other articles for processing, annotate existing documents, post directly to the system, and flag messages for others to see. Multiple information sources in multiple languages are automatically captured, filtered, translated, summarized, and categorized into searchable newsgroups based on disease, region, information source, person, and organization. Critical information is automatically extracted and tagged to facilitate browsing, searching, and sorting.[22]

3.2.8.4.2 CyberTrans

Used by MiTAP, the CyberTrans machine translation system "wraps" either commercial or research translation engines and presents a common set of interfaces to translate the messages automatically into English.[23]

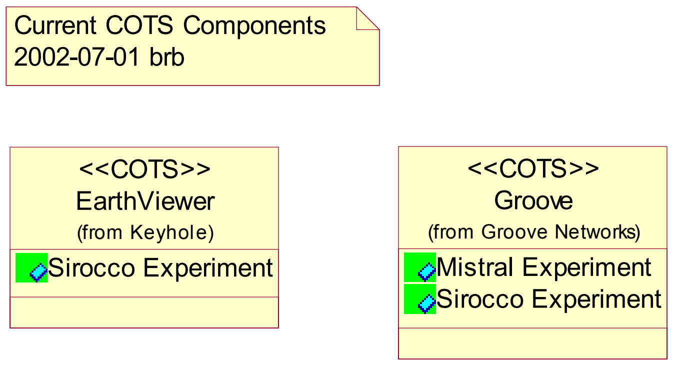
[21] More information about the TIDES program is available at http://www.darpa.mil/iao/TIDES.htm.

[22] Damianos, L. Day, D. Hirschman, L. et. al. "Real Users, Real Data, Real Problems: The MiTAP System for Monitoring Bio Events," *Proceedings of BTR2002: Unified Science & Technology for Reducing Biological Threats & Countering Terrorism*, The University of New Mexico, March 2002.

[23] Miller, K., Reeder, F., Hirschman, L., Palmer, D. (2001). "Multilingual Processing for Operational Users," *NATO Workshop on Multilingual Processing at EUROSPEECH*, September 2001.

3.2.8.5 Commercial Off The Shelf Software (COTS)

This section captures the COTS software that is being used explored for experimentation.



3.2.8.5.1 EarthViewer

EarthViewer fuses high-resolution satellite and aerial imagery, elevation data, GPS coordinates, and overlay information on cities and businesses to deliver a streaming three-dimensional map of the globe.[24]

3.2.8.5.2 Groove

Groove Workspace provides a peer-to-peer, secure, collaborative environment for sharing files and creating applications.[25]

3.2.8.5.3 Open Source

This section captures the Open Source software that is being used explored for experimentation.

[24] More information about EarthViewer is available at http://www.earthviewer.com/.

[25] More information about Groove is available at http://www.groove.net/.

3.2.9. System Implementation View

The following is the planned release of functionality of the TIA system described above. This initial set of functionality is focused on setting up the supporting application infrastructures such as core services, TIA Portal, and the round-trip integration between the multiple platforms. This section will be updated as additional functionality and Application Services are identified through experimentation for incorporation into the TIA System.

Release 1 : Base Portal **1 Jul 02**

- Base Portal capabilities such as Login, Display of portlets, and foundation security model
- Include Portal categories of "Welcome," "Corporate Memory," and "Integrated Tools"
- Embedded document repository with integrated search
- Initial publish Data Services interface to the Portal corporate memory-document repository

Release 2: Groove - Portal Integration **22 Jul 02**

- Enhanced round trip interaction between Portal and Groove platforms.
- Tighter integration between Portal and Groove — spawn a shared space from Portal activity such as MiTap search.
- Within TIA enhanced Groove files tool, an end user can drag and drop files of the shared space to a specific folder within the Portal document repository.

Release 3: Integrated Security Services **26 Aug 02**

- Consolidated integration of Portal and Groove with Core Services security framework
- Incorporate additional content in TIA Portal — TIA news, TIA general info, etc.

Release 4: Initial Application Service Integration **TBD**

- Identify and integrate application within TIA Portal (Web based) or Groove

3.2.9.1 Multi-Level Security Approach

Due to the multiple networks involved in the TIA system, Multi Level Security (MLS) guard technology is needed to bridge the classification of networks and exchange releasable information. There are several guard technologies in existence today such as Getronics Command and Control Guard (C2G), ISSE guard, and Trusted Computing Solutions WebGuard product. One of the most advanced in this area is the NetTop [7] platform architecture developed by the National Security Agency. NetTop stands for a Network on your Desktop. Based on Virtual Machine Monitor (VMM) technology, Vmware, it resides on a trusted Linux operating system and isolates the operating system from the client application. This architecture enables multiple applications to coexist on the same hardware platform communicating in a TCP/IP loopback mode, or virtual Ethernet. Using the NetTop architecture, multiple configurations can be assembled and applied to the TIA system. The primary solution is for interconnecting different classifications of networks. Another area of application is to use NetTop as a trusted platform for TIA applications to reside on. This can provide an additional security measure for system accreditation ensuring that no rogue applications can access OS resources to perform malicious actions.

3.2.10. References

1. Java API for XML-based RPC (JAX-RPC) Specification version 0.8, Published on March 5, 2002 by Sun Microsystems, Inc. http://java.sun.com/xml/jaxrpc/index.html

2. Open Grid Services Architecture (OGSA) Draft Specification, Published on Feb. 15, 2002 by S. Tuecke et all. http://www.globus.org/research/papers/gsspec.pdf

3. Simple Object Access Protocol (SOAP) 1.1 Specification, Published on May 8, 2000 by W3C. http://www.w3.org/TR/SOAP

4. Web Services Definition Language (WSDL) 1.1 Specification, Published on March 15, 2002 by W3C. http://www.w3.org/TR/wsdl

5. Universal Description, Discovery and Integration (UDDI) 2.0 Specification, Published on June 8, 2002 by UDDI.org. http://uddi.org/specification.html

6. Web Services Security (WS-Security), Published in April 2002 by IBM. http://www-106.ibm.com/developerworks/library/ws-secure

7. NetTop – A Network on your desktop. Tech Trend Notes, Volume 9, Edition 4, Published in Fall 2000 by National Security Agency.

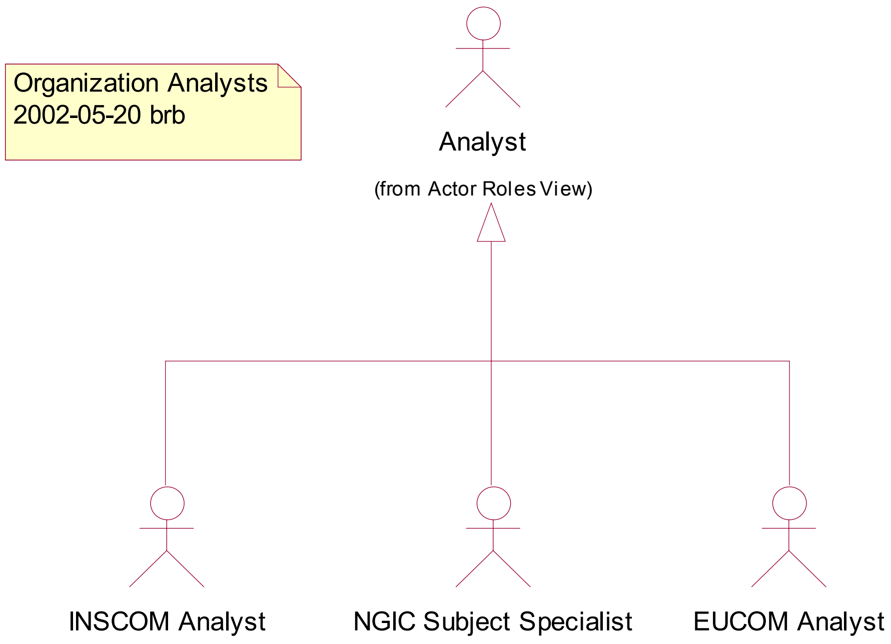
4. Experiment Models

4.1. *Mistral Experiment (2002-05-22)*

4.1.1. Actor Roles View

Three Actors Roles are used in the 22 May 2002 experiments.

- INSCOM Analyst
- NGIC Analyst
- EUCOM Analyst

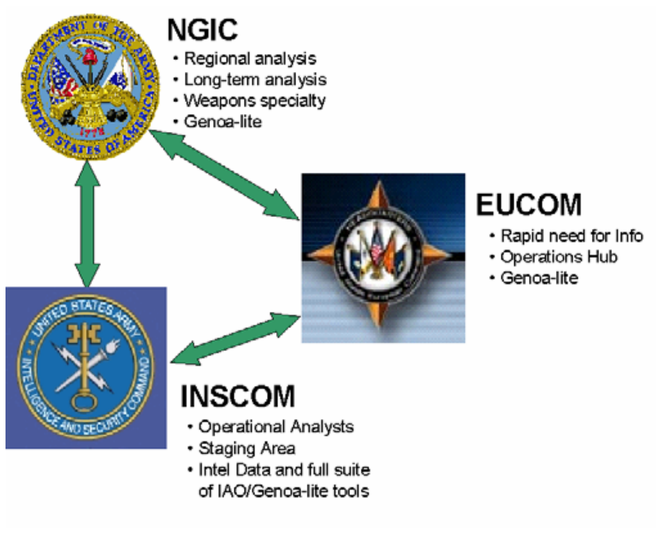


4.1.1.1 INSCOM Analyst

4.1.1.2 EUCOM Analyst

4.1.1.3 NGIC Subject Specialist

4.1.2. Objectives and Results



4.1.2.1 Objectives (Metrics for Success)

4.1.2.1.1 Deploy research and development software creating a collaborative (Groove) environment on operational networks.

4.1.2.1.2 Create a baseline experiment CONOPS to support an end-to-end ongoing experiment process.

4.1.2.1.3 Establish an infrastructure for software, hardware, and people that will serve as the basis for future experiments and future IAC deployments.

4.1.2.1.4 Include operational users early using a real-world problem, leveraging existing real-world data.

4.1.2.1.5 Integrate Genoa, EELD, TIDES, and INSCOM tools and data to create end-to-end functionality.

4.1.2.2 Sources of Potentially New Insights

The operational nature and data being used in the experiment have the potential to create unique insights and contribute to solving a real-world problem.

4.1.2.2.1 New Models of XYZ Group

4.1.2.2.2 Focused search (with search tools)

4.1.2.2.3 Collaboration between analysts

4.1.2.3 Results

4.1.2.3.1 Team among INSCOM, technology providers, analysts, and soldiers.

4.1.2.3.2 Established Groove collaborative environment between INSCOM, NGIC, and 66th MI.

4.1.2.3.3 TACLANE enabled VPN tunneling through JWICS.

4.1.2.3.4 Granted INSCOM-specified IATO.

4.1.2.3.5 NSANet IATO approved.

4.1.2.3.6 Infrastructure leave-behind payoffs.

1. File sharing with classified data over Groove.
2. NGIC SIPRNET data transfer through Groove.
3. Communications through a common CIM Model.
4. Immediate ability for operational personnel to use and test CIM/Verona tools, now permanently installed.
5. CIM and Verona components are of immediate value to the operators.
6. Successful vetting of CIM model with NGIC including spontaneous comments.

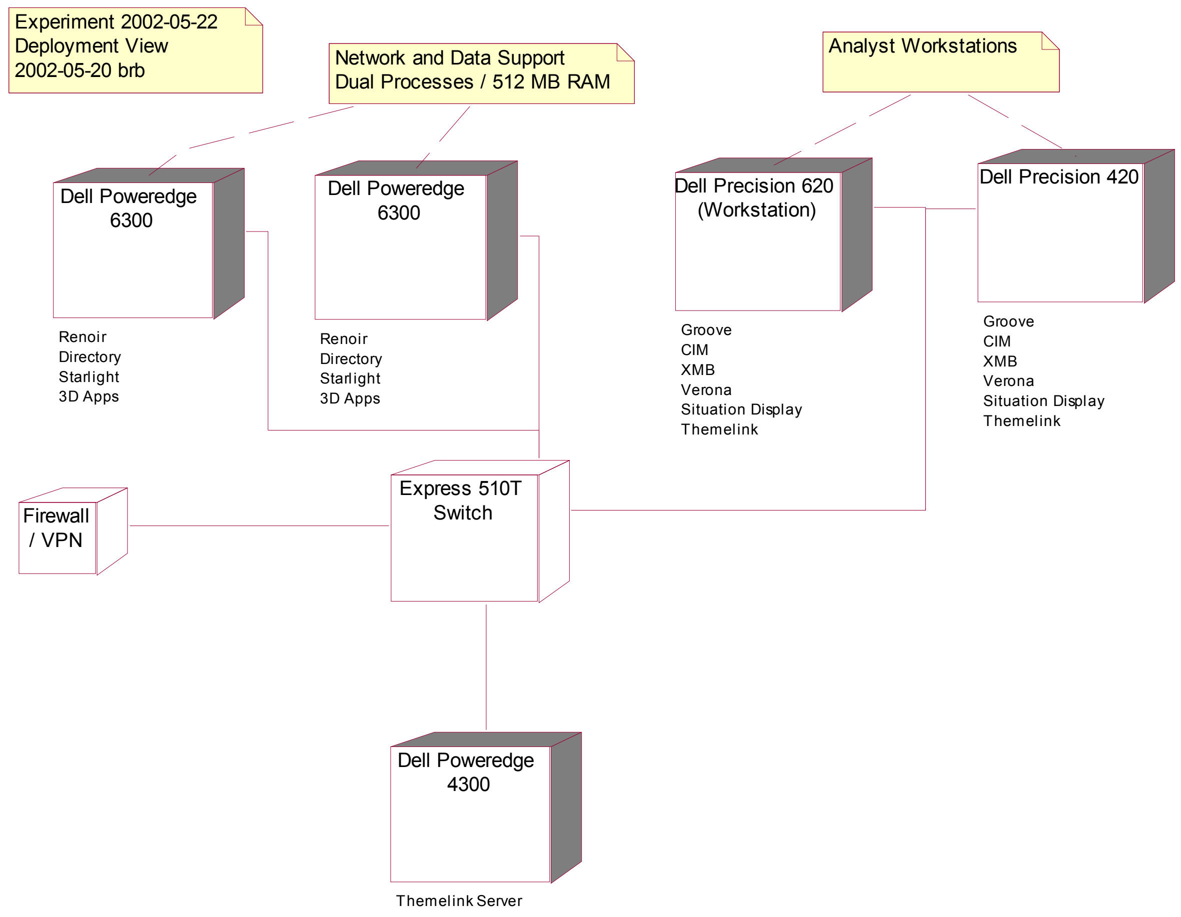
4.1.3. Experiment Deployment View

The Experiment Deployment view describes the mapping(s) of the software, identified in the Business Application Service View and used in Experiments, onto the hardware and reflects its distributed aspect. It is analogous to the Physical View of the Rational 4 + 1 View of System Architecture.

The Experiment Deployment View currently contains two diagrams of the experimental deployment. A UML Deployment diagram describes the hardware and software configuration at the DPP. A System Topology Diagram describes the network connectivity between INSCOM and the other experimental sites.

4.1.3.1 UML Deployment Diagram

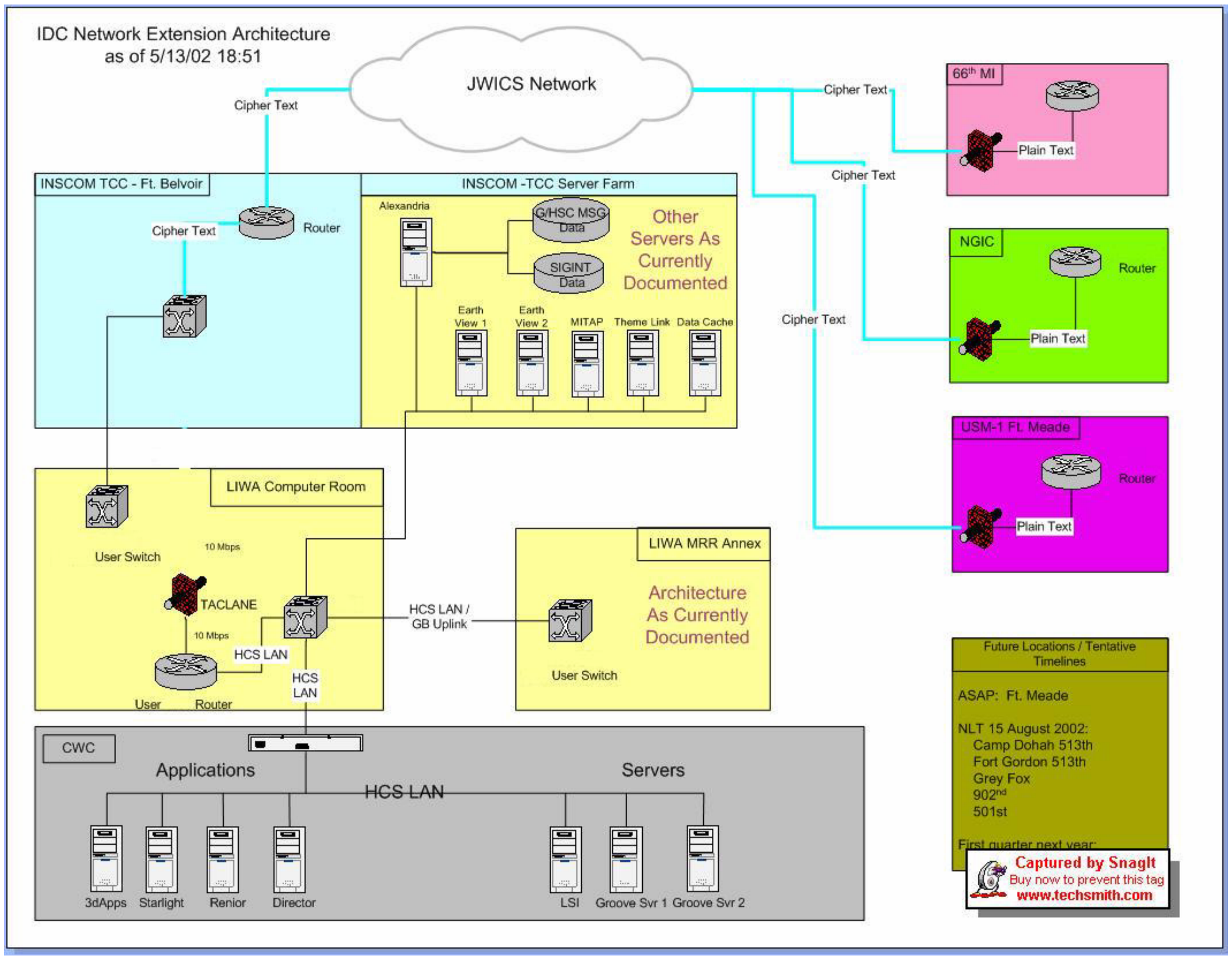
The UML Deployment diagram shows the hardware and software configuration at the Springfield Facility (DPP) and identifies individual packages running on workstation and server machines.



Mistral Experiment Deployment Diagram

4.1.3.2 System Topology Diagram

The System Topology Diagram shows the server infrastructure and network topology used to connect the three sites, INSCOM, NGIC, and EUCOM in the 22 May 2002 experiments.



Mistral Experiment System Topology Diagram

4.1.3.3 Business Information View

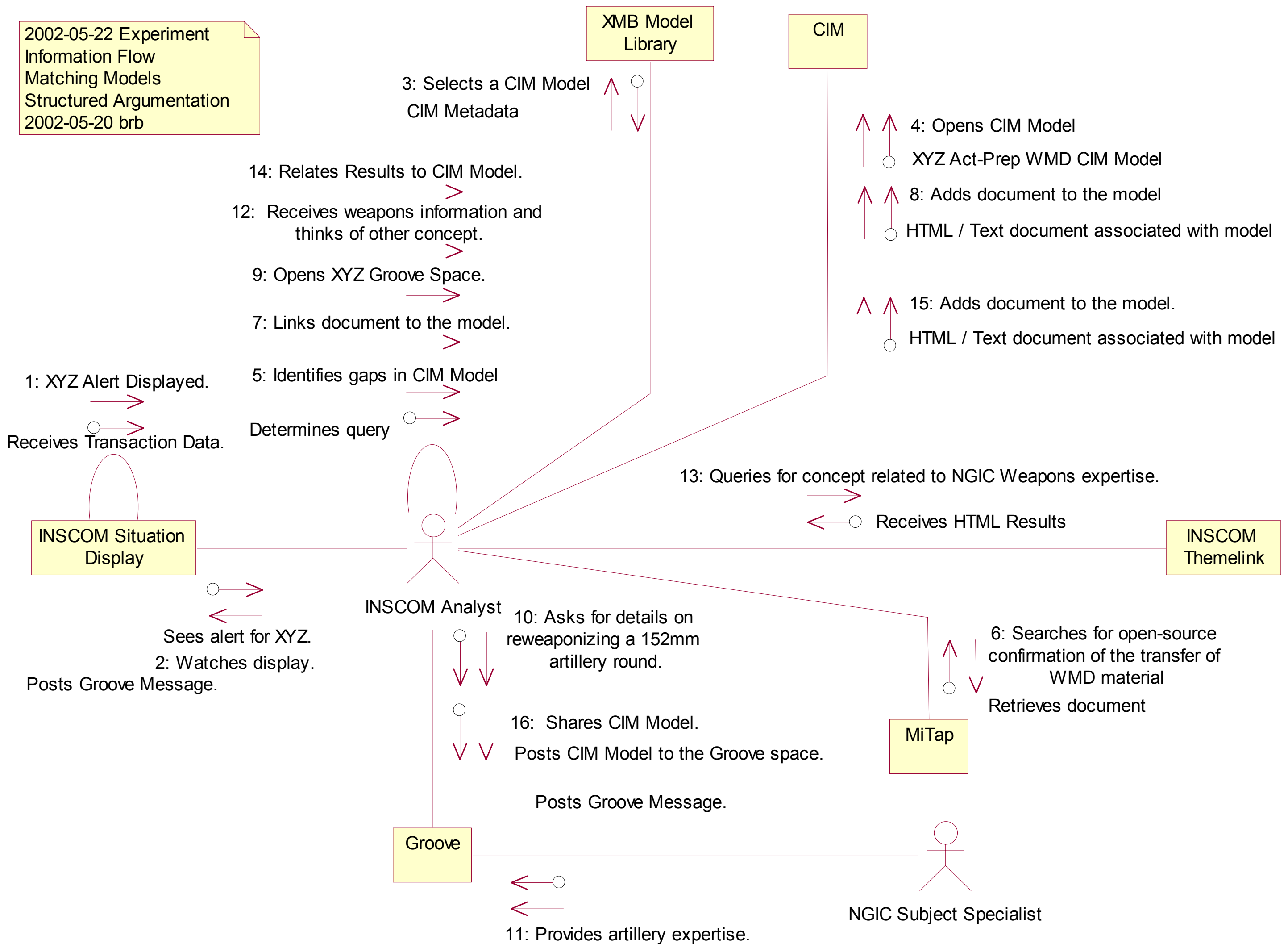
The Business Information View for the experiment describes:

- The products (formats) used in the experiment.
- The flow of roles, technology components, and information within the experiment.
- Files used by the technology components in the experiment.
- The work flows of the experiment from an information perspective.

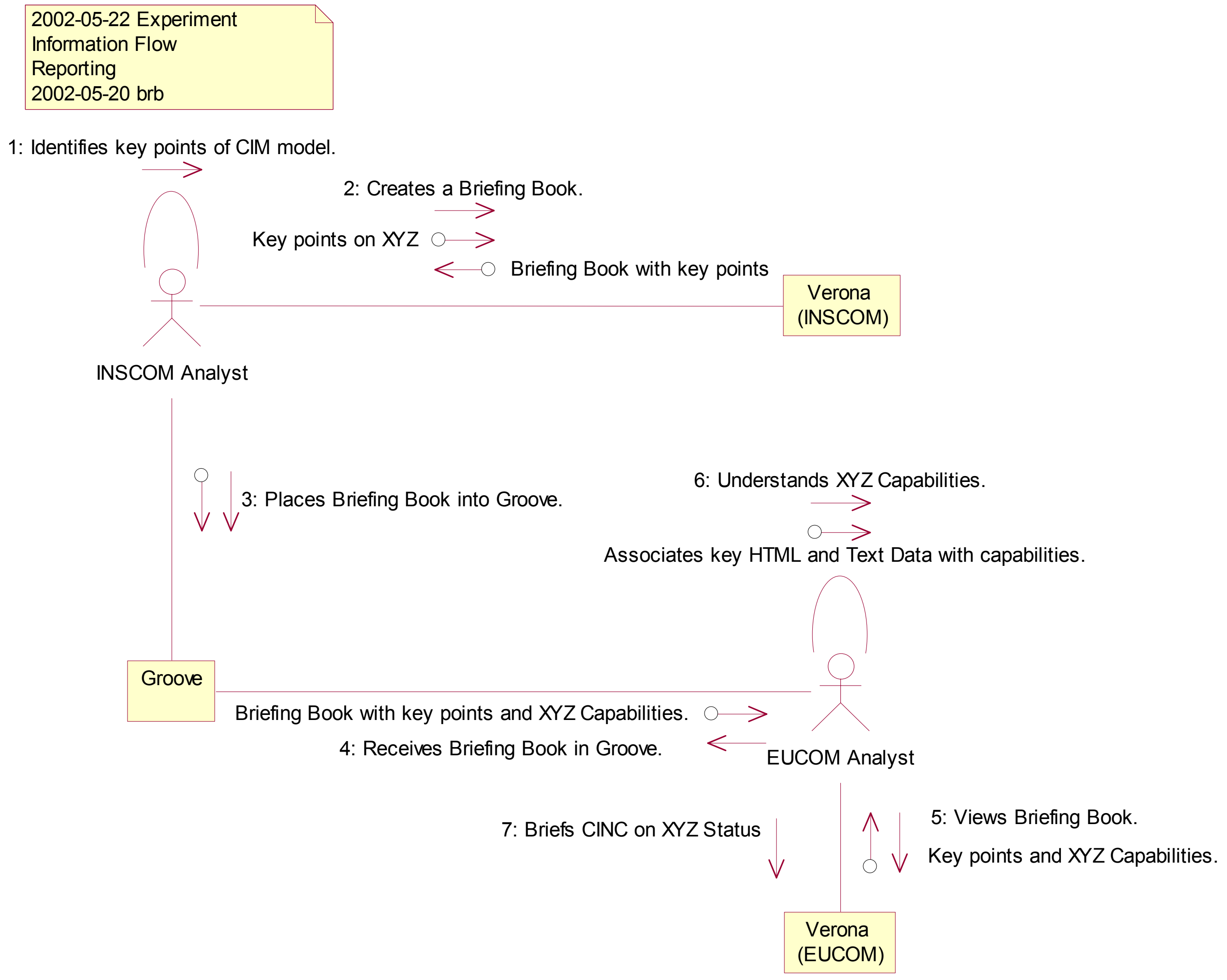
4.1.3.3.1 Information Flow

The flow for the experiment is partitioned into three diagrams, each corresponding to a set of the use cases exercised in the 22 May 2002 experiment. The collaboration diagrams represent the interactions between the technology components and Actor roles. In addition, there is a data-flow arrow indicating the information product that is being exchanged, modified, and/or used at each stage.

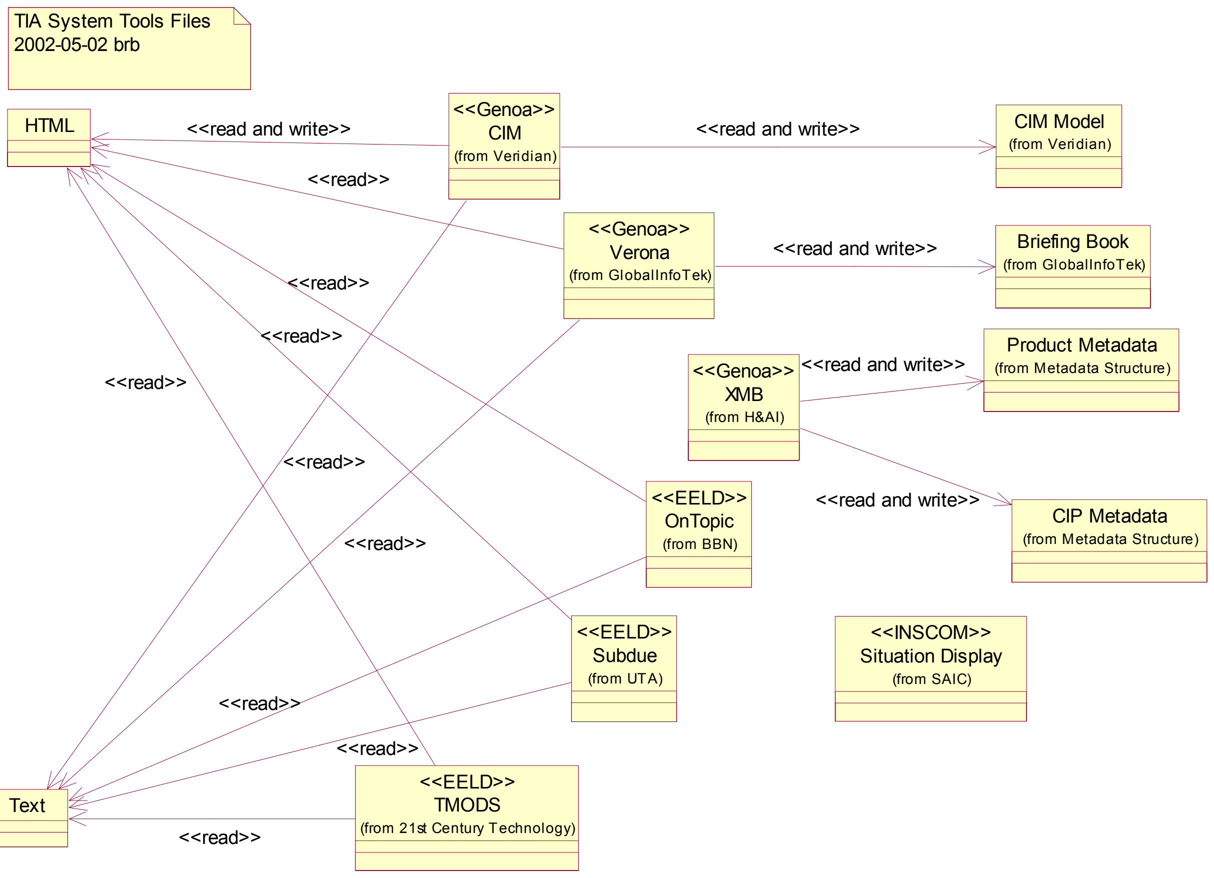
4.1.3.3.1.1 Matching Models and Structured Argumentation


2002-05-22 Experiment
Information Flow
Matching Models
Structured Argumentation
2002-05-20 brb
XMB Model Library
CIM
3: Selects a CIM Model
CIM Metadata
4: Opens CIM Model
XYZ Act-Prep WMD CIM Model
8: Adds document to the model
HTML / Text document associated with model
14: Relates Results to CIM Model.
12: Receives weapons information and thinks of other concept.
9: Opens XYZ Groove Space.
7: Links document to the model.
15: Adds document to the model.
HTML / Text document associated with model
1: XYZ Alert Displayed.
Receives Transaction Data.
5: Identifies gaps in CIM Model
Determines query
13: Queries for concept related to NGIC Weapons expertise.
Receives HTML Results
INSCOM Situation Display
INSCOM Themelink
INSCOM Analyst
Sees alert for XYZ.
2: Watches display.
Posts Groove Message.
10: Asks for details on reweaponizing a 152mm artillery round.
6: Searches for open-source confirmation of the transfer of WMD material
Retrieves document
16: Shares CIM Model.
Posts CIM Model to the Groove space.
MiTap
Posts Groove Message.
Groove
NGIC Subject Specialist
11: Provides artillery expertise.

4.1.3.3.1.2 Reporting


2002-05-22 Experiment
Information Flow
Reporting
2002-05-20 brb
1: Identifies key points of CIM model.
2: Creates a Briefing Book.
Key points on XYZ
Briefing Book with key points
Verona (INSCOM)
INSCOM Analyst
3: Places Briefing Book into Groove.
6: Understands XYZ Capabilities.
Associates key HTML and Text Data with capabilities.
Groove
Briefing Book with key points and XYZ Capabilities.
4: Receives Briefing Book in Groove.
EUCOM Analyst
7: Briefs CINC on XYZ Status
5: Views Briefing Book.
Key points and XYZ Capabilities.
Verona (EUCOM)

4.1.3.3.1.3 Information Products


TIA System Tools Files
2002-05-02 brb
HTML
<<Genoa>> CIM (from Veridian)
CIM Model (from Veridian)
<<Genoa>> Verona (from GlobalInfoTek)
Briefing Book (from GlobalInfoTek)
<<Genoa>> XMB (from H&AI)
Product Metadata (from Metadata Structure)
CIP Metadata (from Metadata Structure)
<<EELD>> OnTopic (from BBN)
<<EELD>> Subdue (from UTA)
<<INSCOM>> Situation Display (from SAIC)
<<EELD>> TMODS (from 21st Century Technology)
Text
<<read and write>>
<<read>>

4.1.3.3.1.3.1 CIM Model

4.1.3.3.1.3.2 CIP Metadata

4.1.3.3.1.3.3 Product Metadata

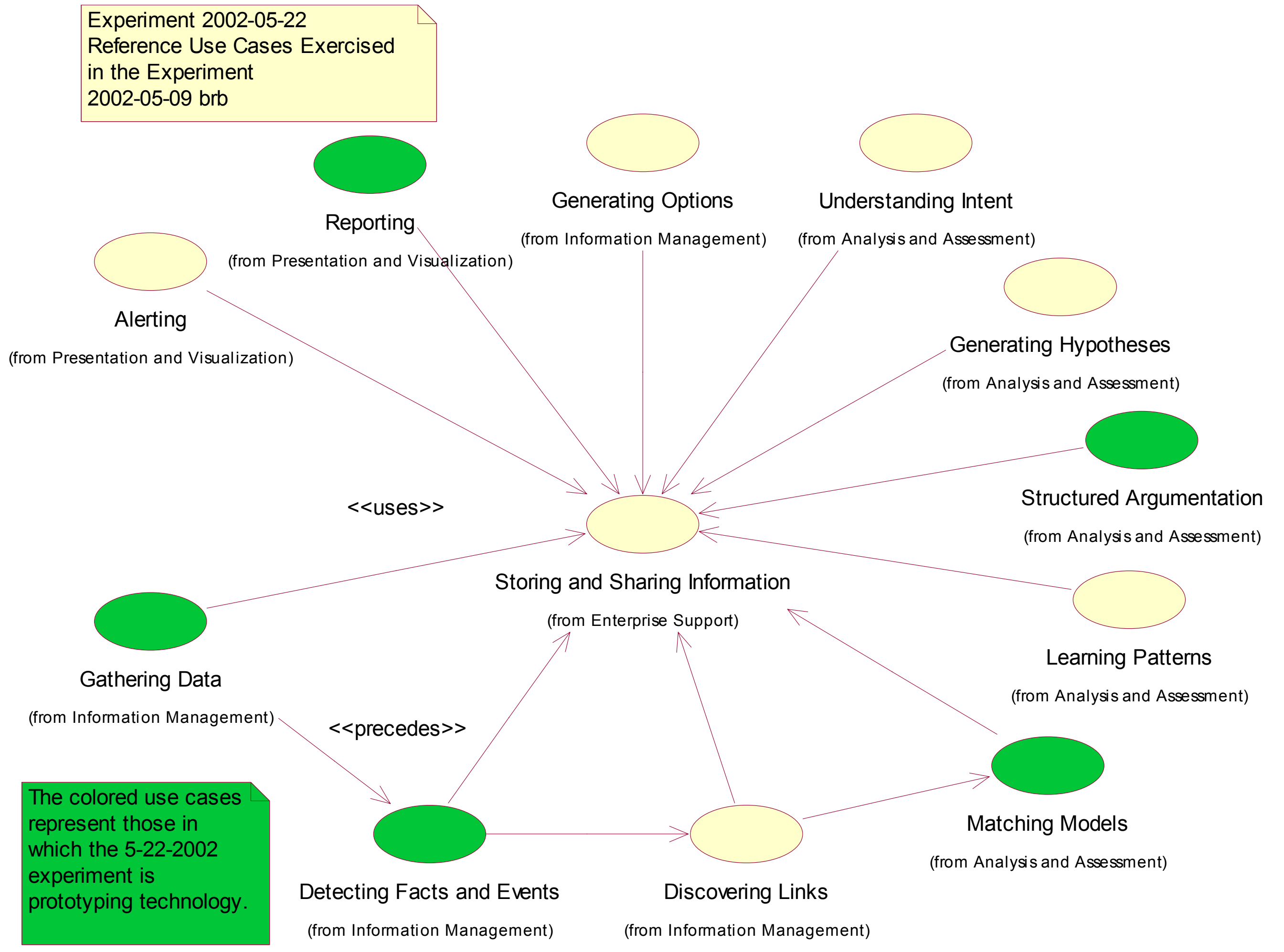
4.1.3.3.1.3.4 HTML Files

4.1.3.3.1.3.5 Text Files

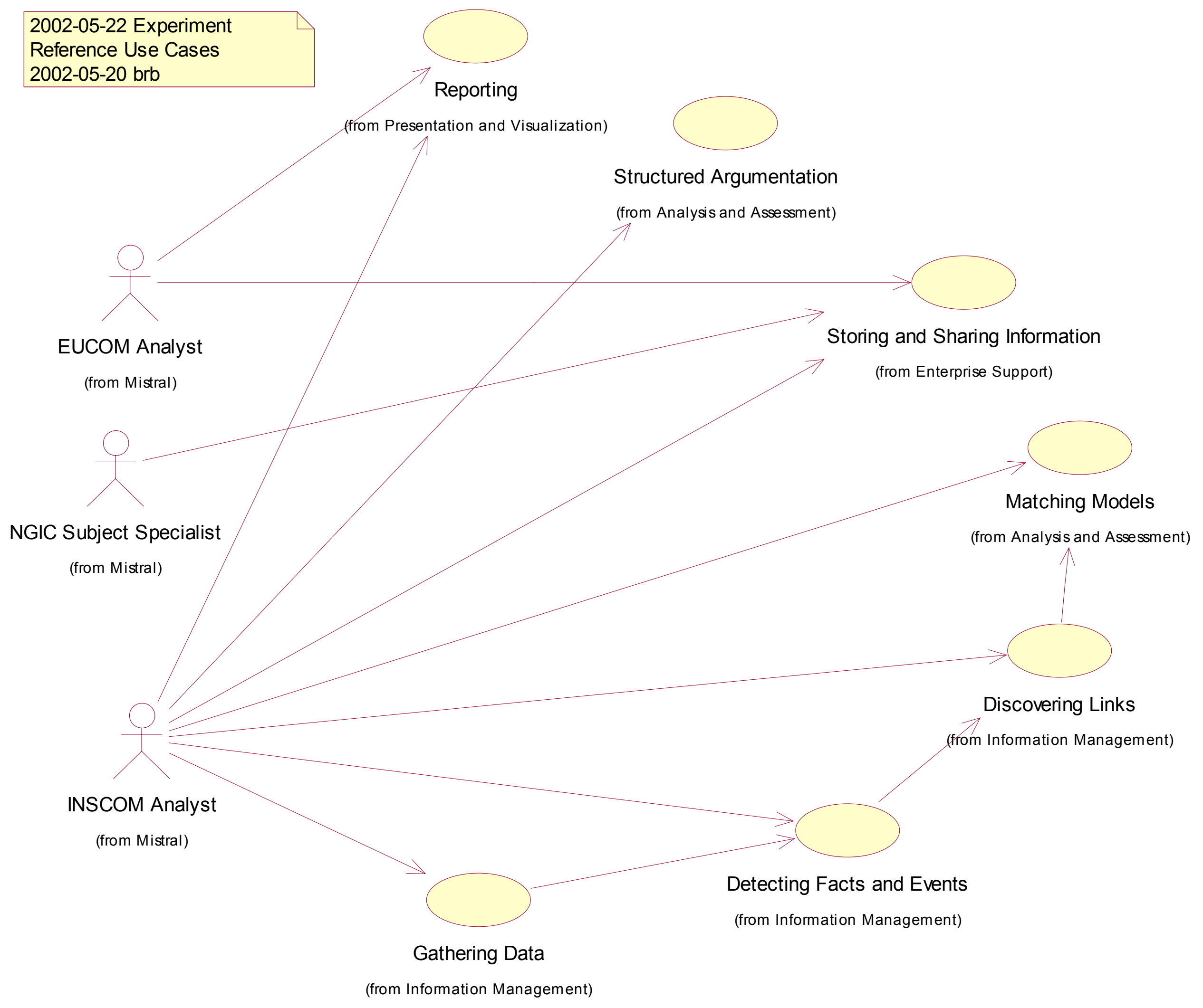
4.1.3.3.1.3.6 Briefing Book

4.1.4. Business Process View

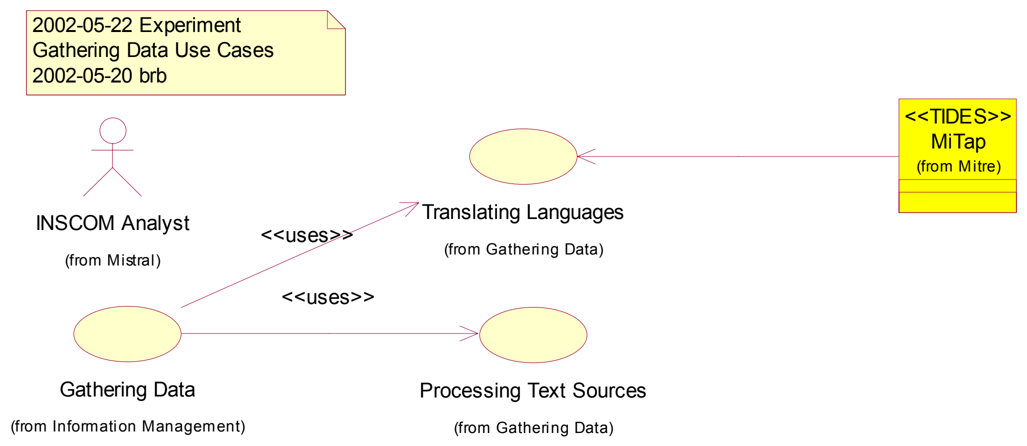
The 22 May 2002 experiment includes capabilities in seven use cases of the TIA Business Model. Use Cases colored in Green indicate their inclusion in the 22 May 2002 experiment.


Experiment 2002-05-22
Reference Use Cases Exercised
in the Experiment
2002-05-09 brb
Reporting
(from Presentation and Visualization)
Generating Options
(from Information Management)
Understanding Intent
(from Analysis and Assessment)
Alerting
(from Presentation and Visualization)
Generating Hypotheses
(from Analysis and Assessment)
Structured Argumentation
(from Analysis and Assessment)
<<uses>>
Storing and Sharing Information
(from Enterprise Support)
Gathering Data
(from Information Management)
Learning Patterns
(from Analysis and Assessment)
<<precedes>>
Matching Models
(from Analysis and Assessment)
The colored use cases represent those in which the 5-22-2002 experiment is prototyping technology.
Detecting Facts and Events
(from Information Management)
Discovering Links
(from Information Management)

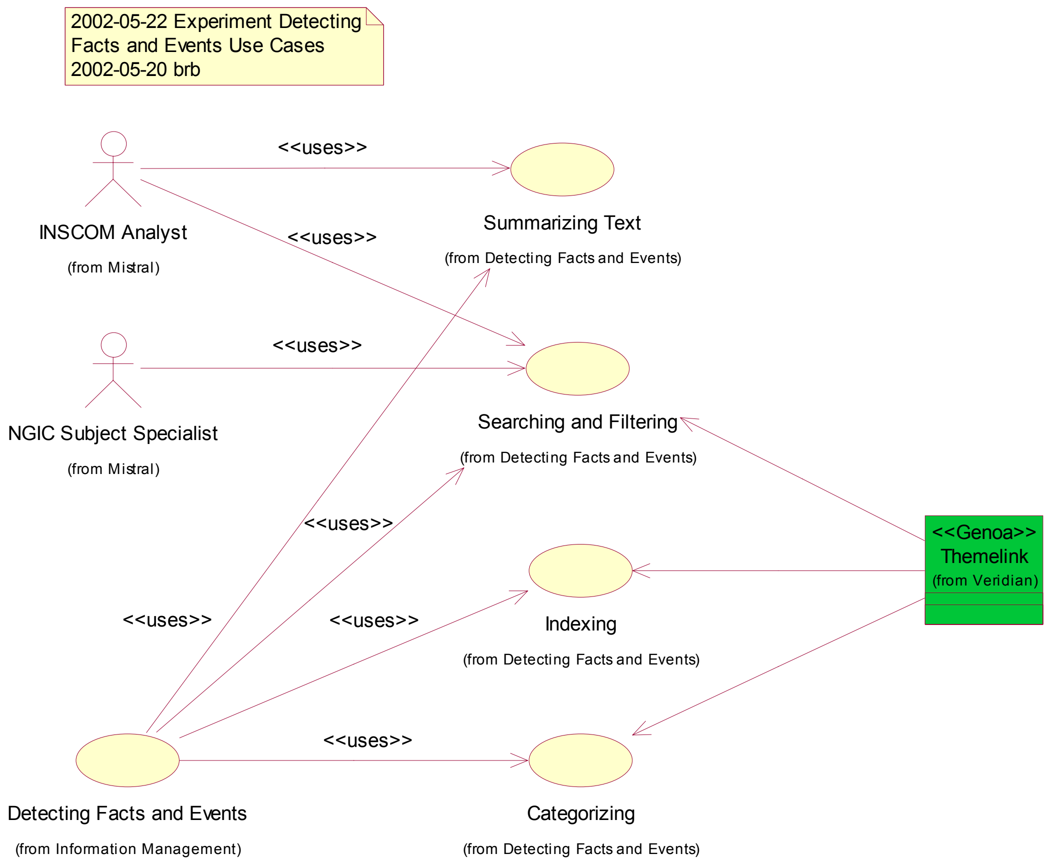
4.1.4.1 Reference Use Cases


2002-05-22 Experiment
Reference Use Cases
2002-05-20 brb
Reporting
(from Presentation and Visualization)
Structured Argumentation
(from Analysis and Assessment)
EUCOM Analyst
(from Mistral)
Storing and Sharing Information
(from Enterprise Support)
NGIC Subject Specialist
(from Mistral)
Matching Models
(from Analysis and Assessment)
Discovering Links
(from Information Management)
INSCOM Analyst
(from Mistral)
Detecting Facts and Events
(from Information Management)
Gathering Data
(from Information Management)

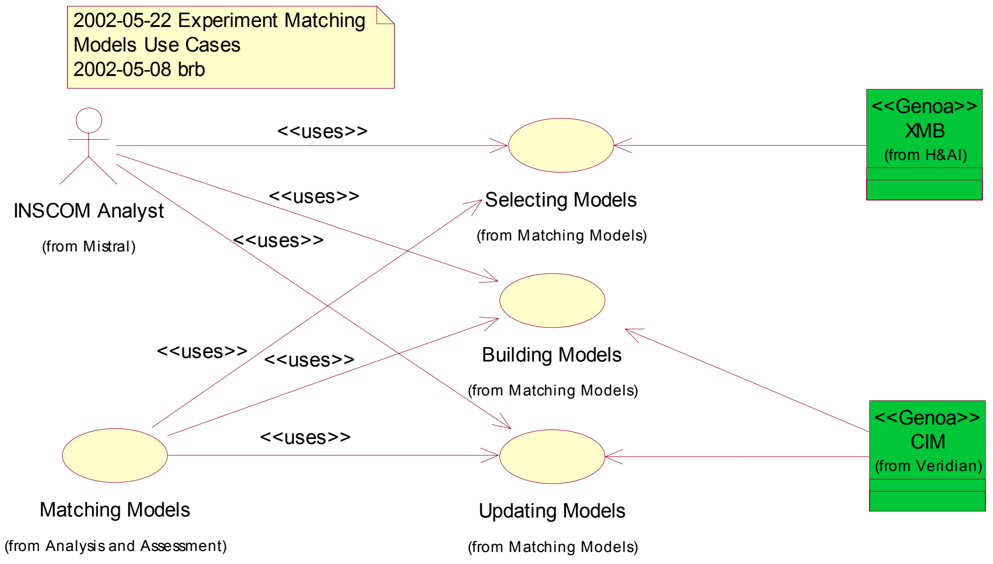
4.1.4.2 Gathering Data



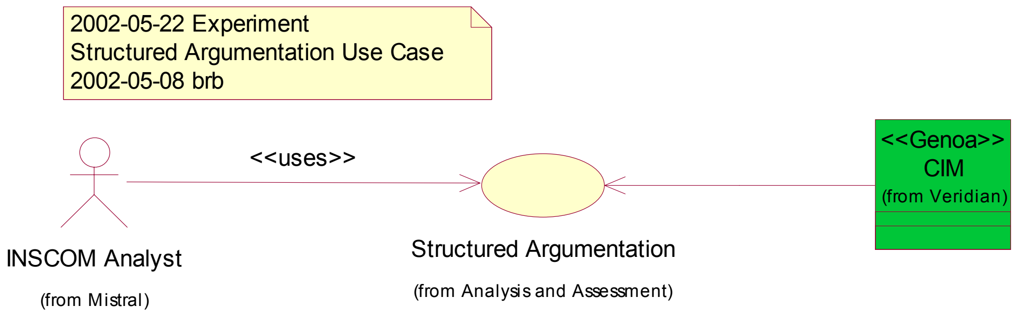
4.1.4.3 Detecting Facts and Events


2002-05-22 Experiment Detecting
Facts and Events Use Cases
2002-05-20 brb
<<uses>>
INSCOM Analyst
(from Mistral)
<<uses>>
Summarizing Text
(from Detecting Facts and Events)
<<uses>>
NGIC Subject Specialist
(from Mistral)
Searching and Filtering
(from Detecting Facts and Events)
<<uses>>
<<Genoa>>
Themelink
(from Veridian)
<<uses>>
<<uses>>
Indexing
(from Detecting Facts and Events)
<<uses>>
Detecting Facts and Events
(from Information Management)
Categorizing
(from Detecting Facts and Events)

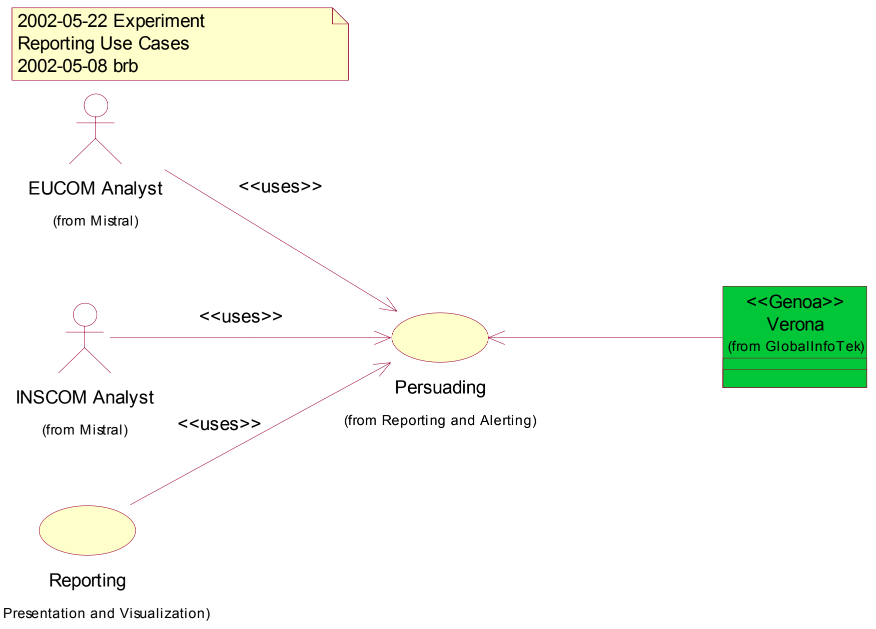
4.1.4.4 Matching Models



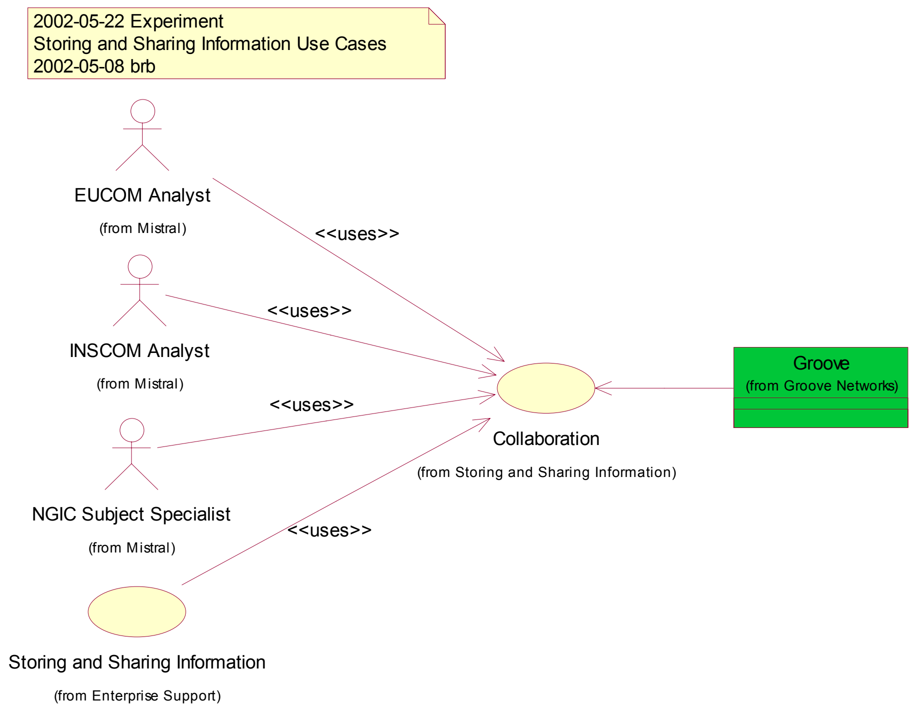
4.1.4.5 Structured Argumentation



4.1.4.6 Reporting



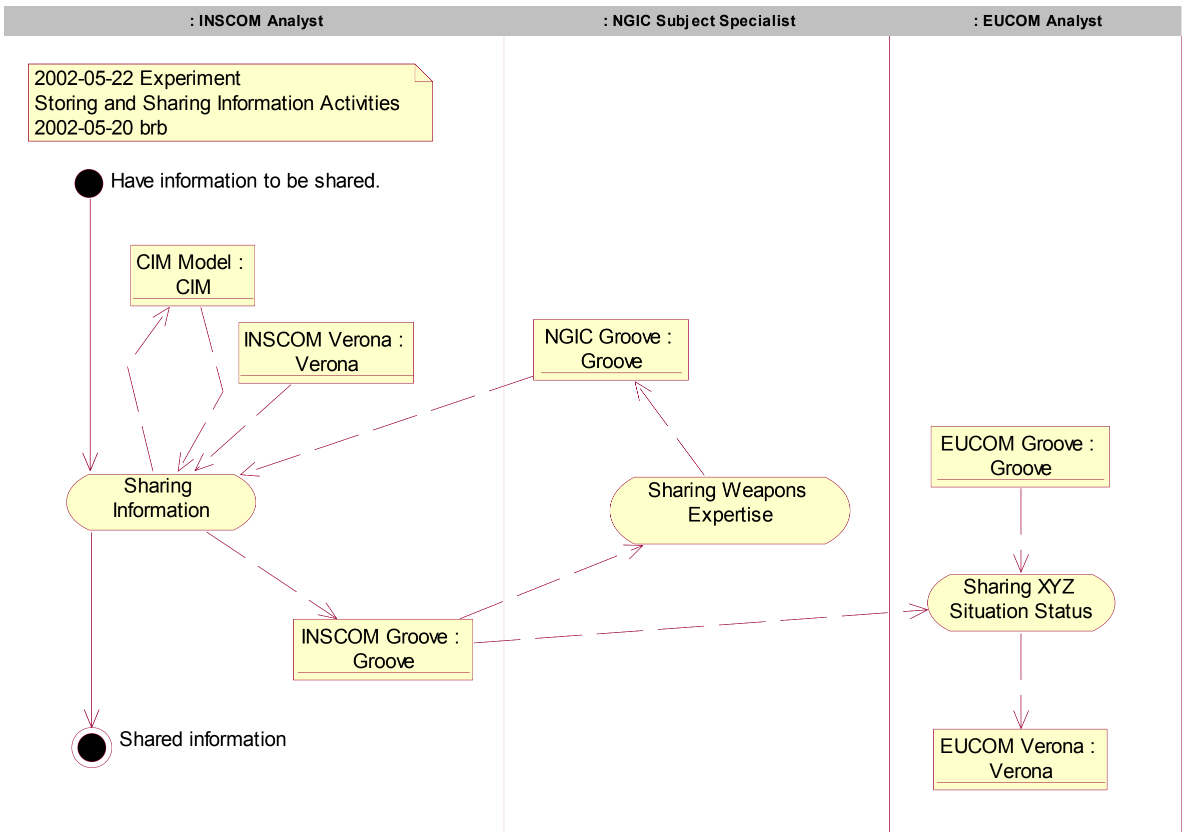
4.1.4.7 Storing and Sharing Information



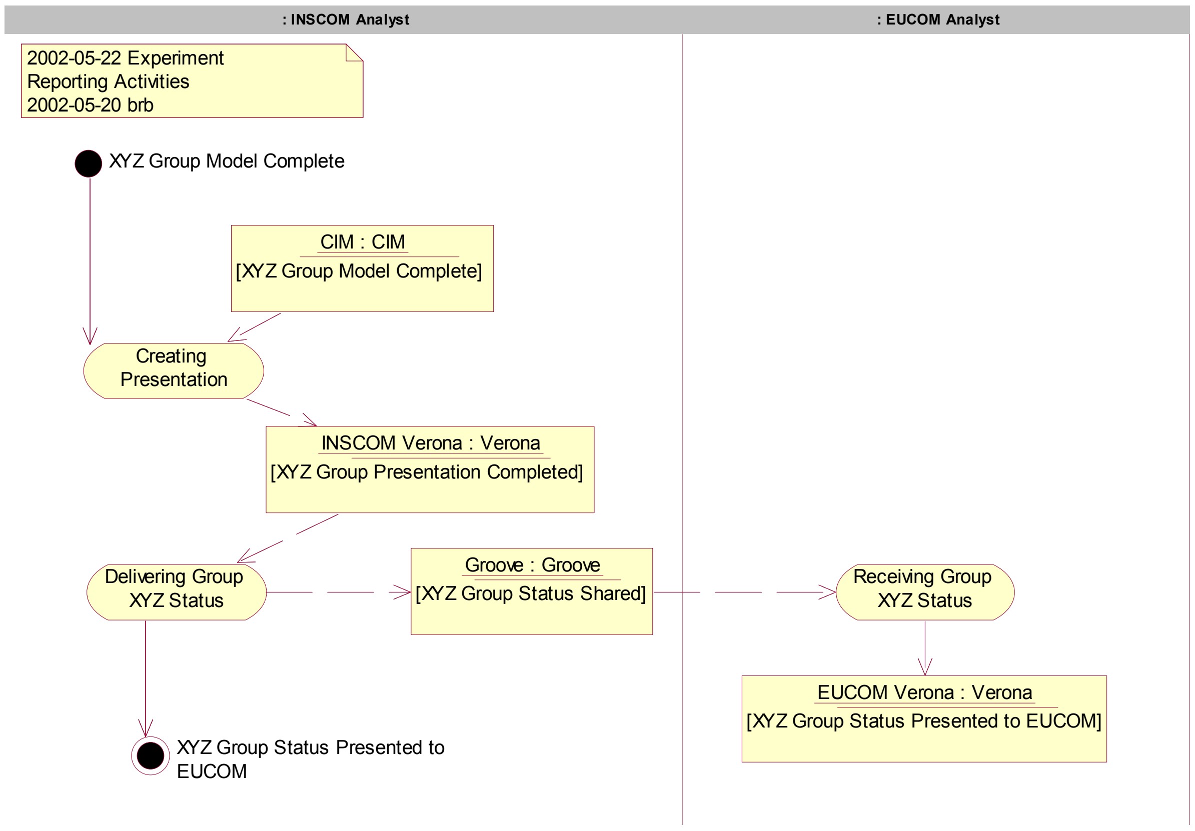
4.1.4.8 Activity Diagrams

The Activity Diagrams depict the specific interactions occurring in the Mistral experiment within each Use Case exercised.

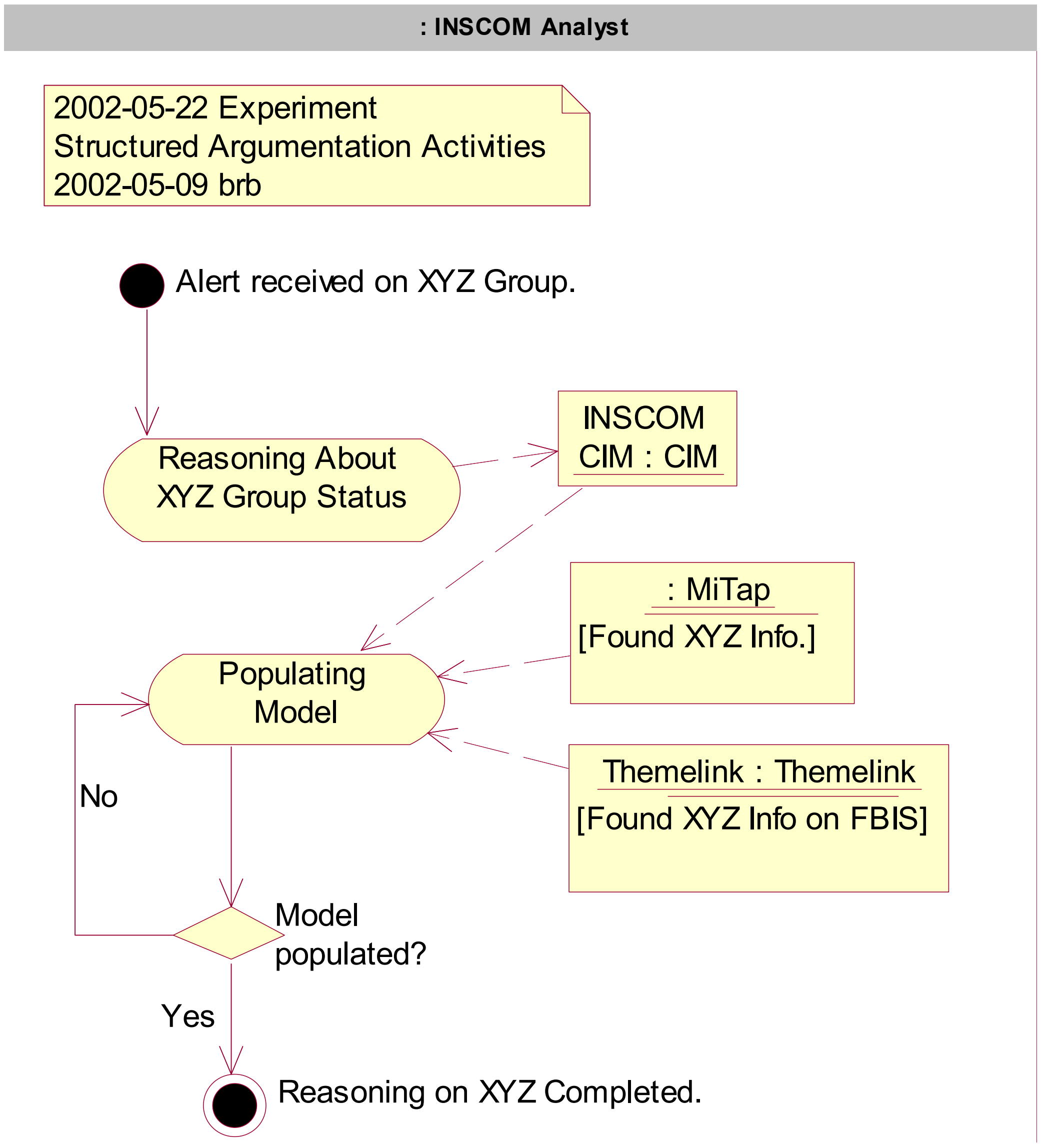
4.1.4.8.1 Sharing Activities



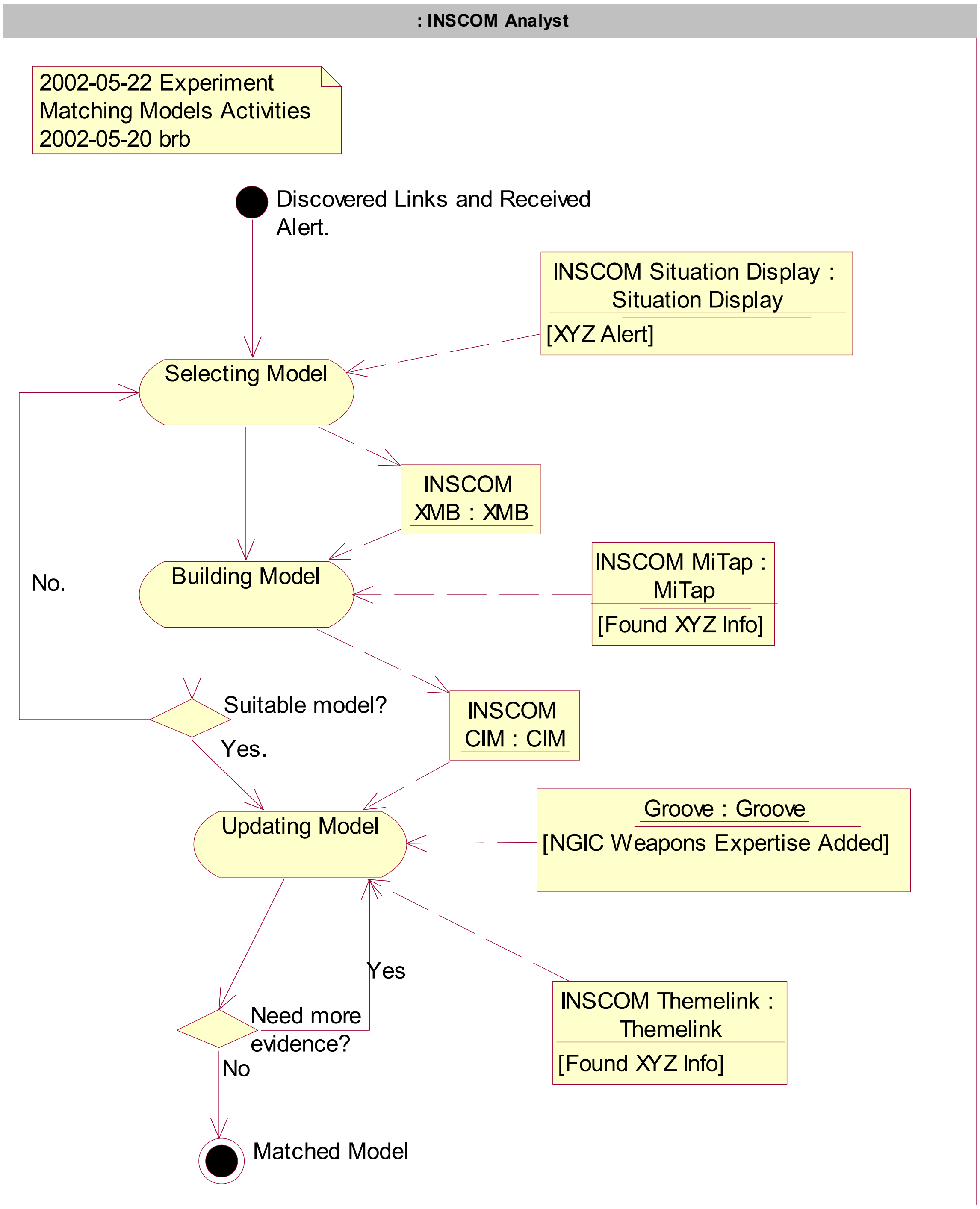
4.1.4.8.2 Reporting Activities


: INSCOM Analyst
: EUCOM Analyst
2002-05-22 Experiment
Reporting Activities
2002-05-20 brb
XYZ Group Model Complete
CIM : CIM
[XYZ Group Model Complete]
Creating Presentation
INSCOM Verona : Verona
[XYZ Group Presentation Completed]
Delivering Group XYZ Status
Groove : Groove
[XYZ Group Status Shared]
Receiving Group XYZ Status
EUCOM Verona : Verona
[XYZ Group Status Presented to EUCOM]
XYZ Group Status Presented to EUCOM

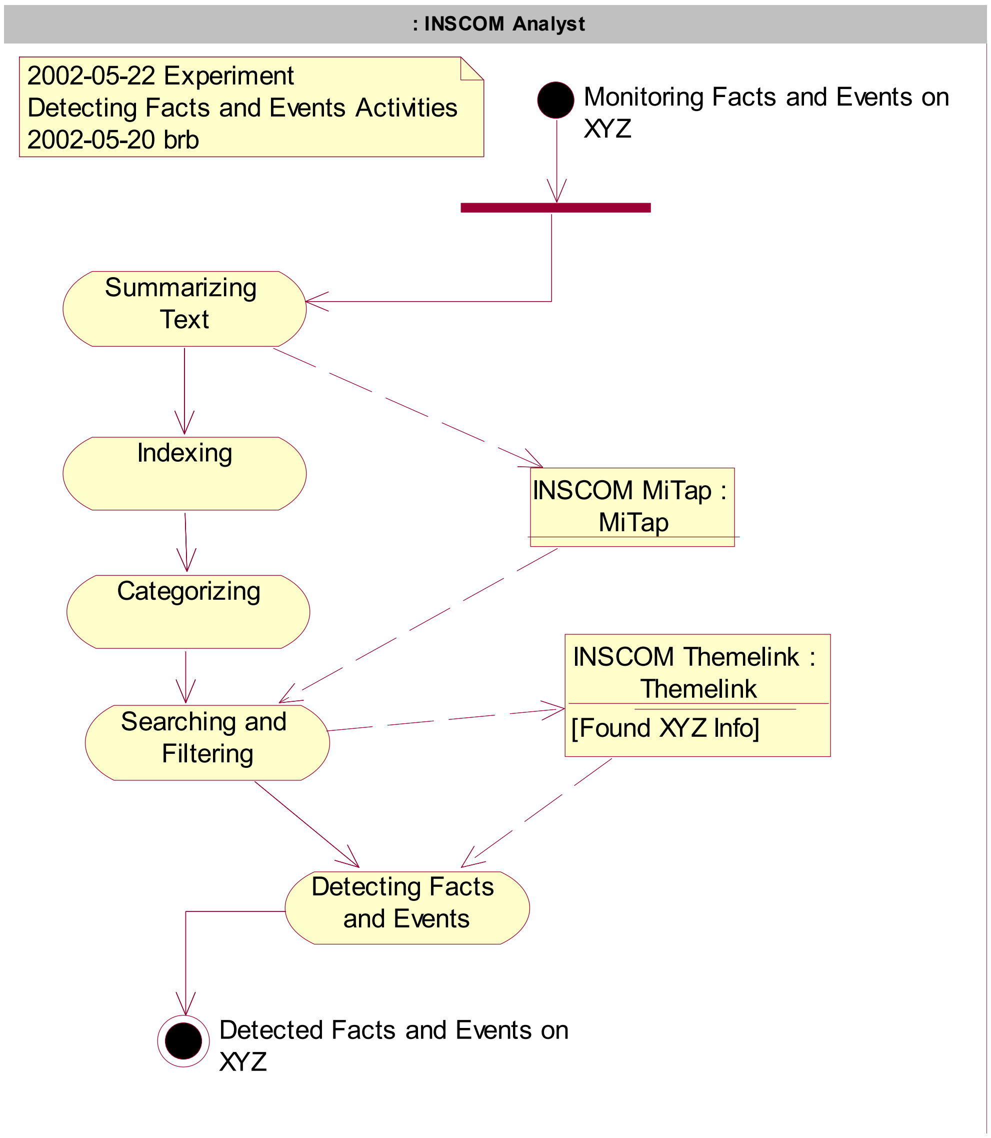
4.1.4.8.3 Structured Argumentation



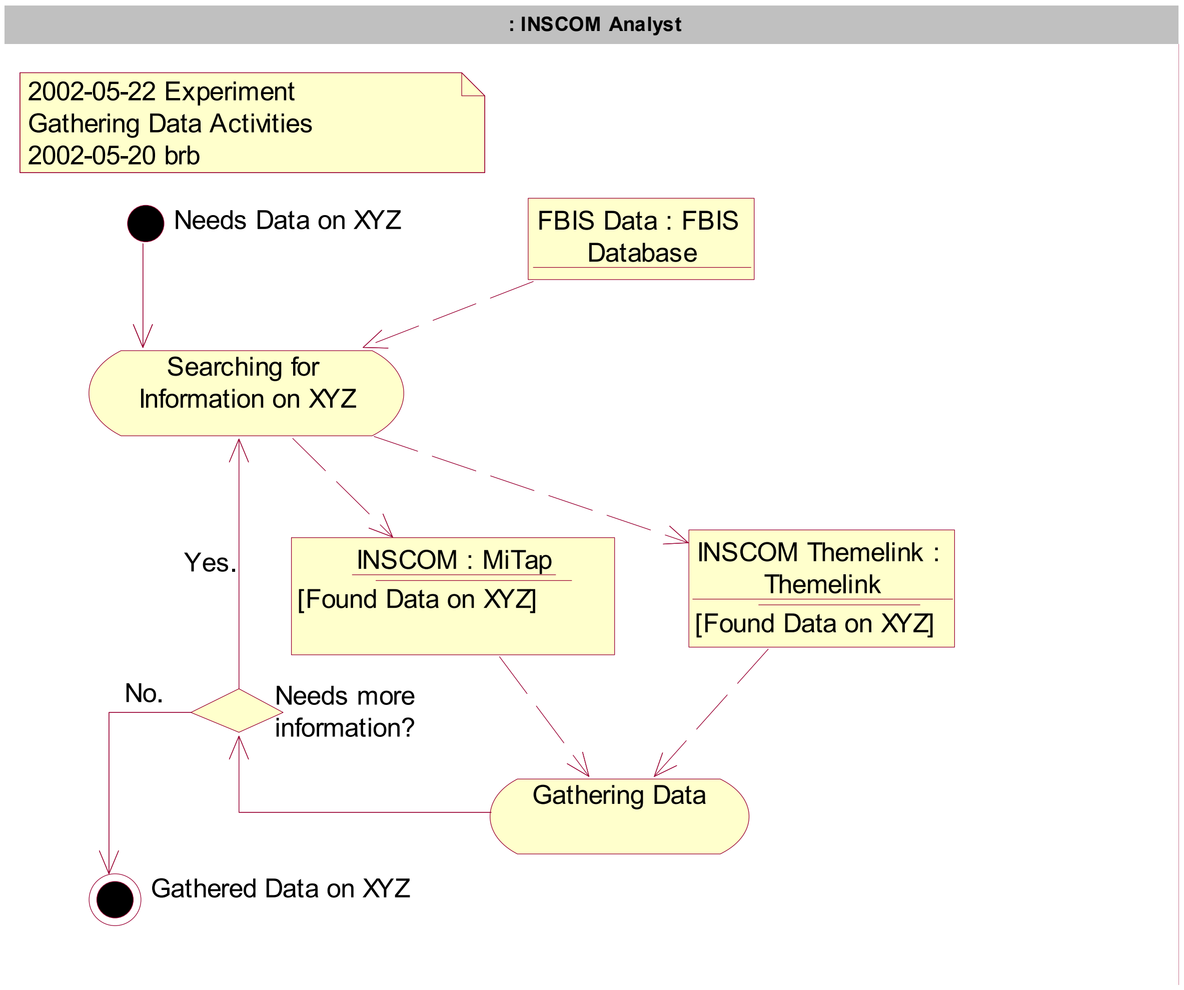
4.1.4.8.4 Matching Models



4.1.4.8.5 Detecting Facts and Events Activities


: INSCOM Analyst
2002-05-22 Experiment
Detecting Facts and Events Activities
2002-05-20 brb
Monitoring Facts and Events on XYZ
Summarizing Text
Indexing
Categorizing
Searching and Filtering
INSCOM MiTap : MiTap
INSCOM Themelink : Themelink
[Found XYZ Info]
Detecting Facts and Events
Detected Facts and Events on XYZ

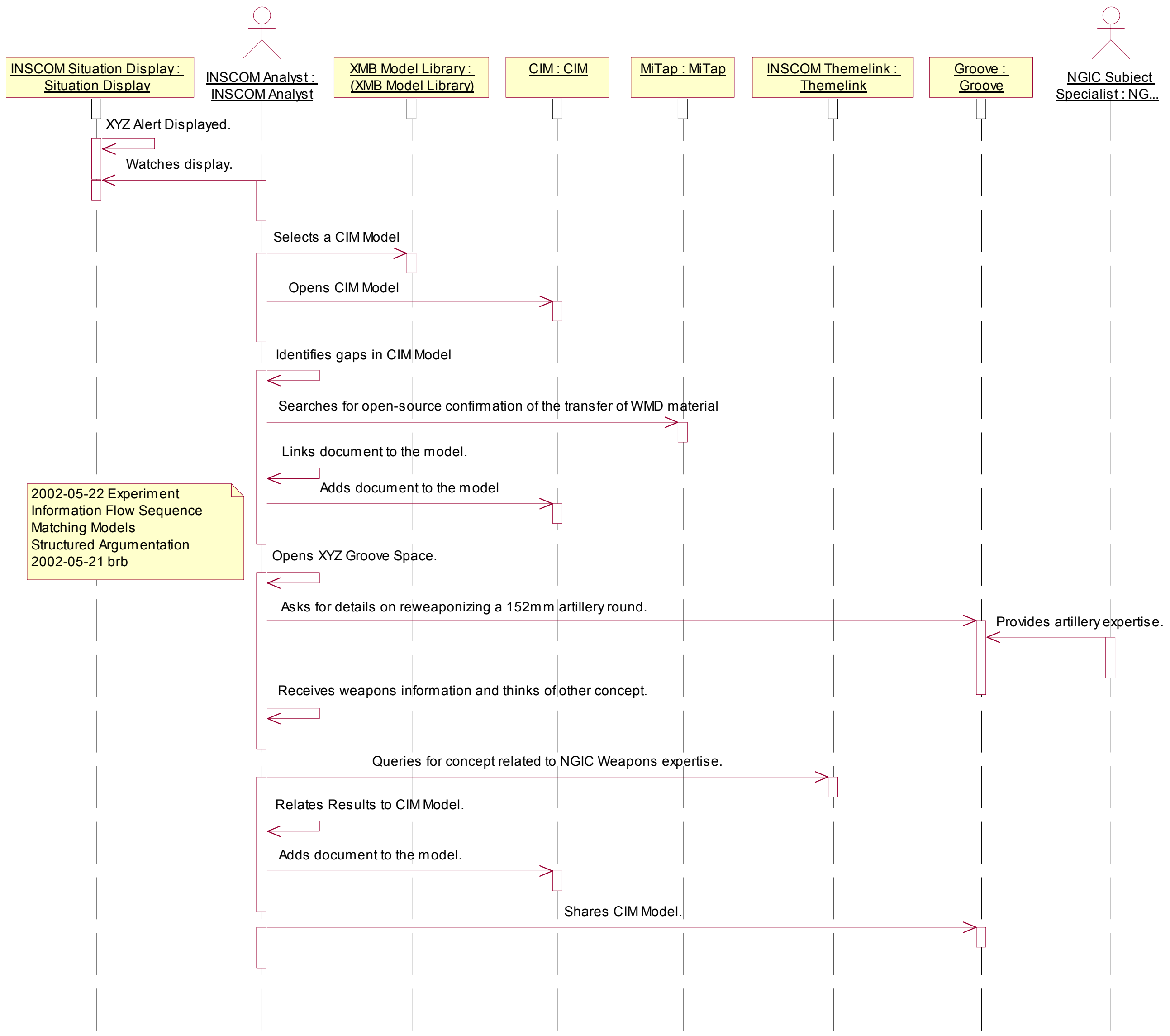
4.1.4.8.6 Gathering Data Activities



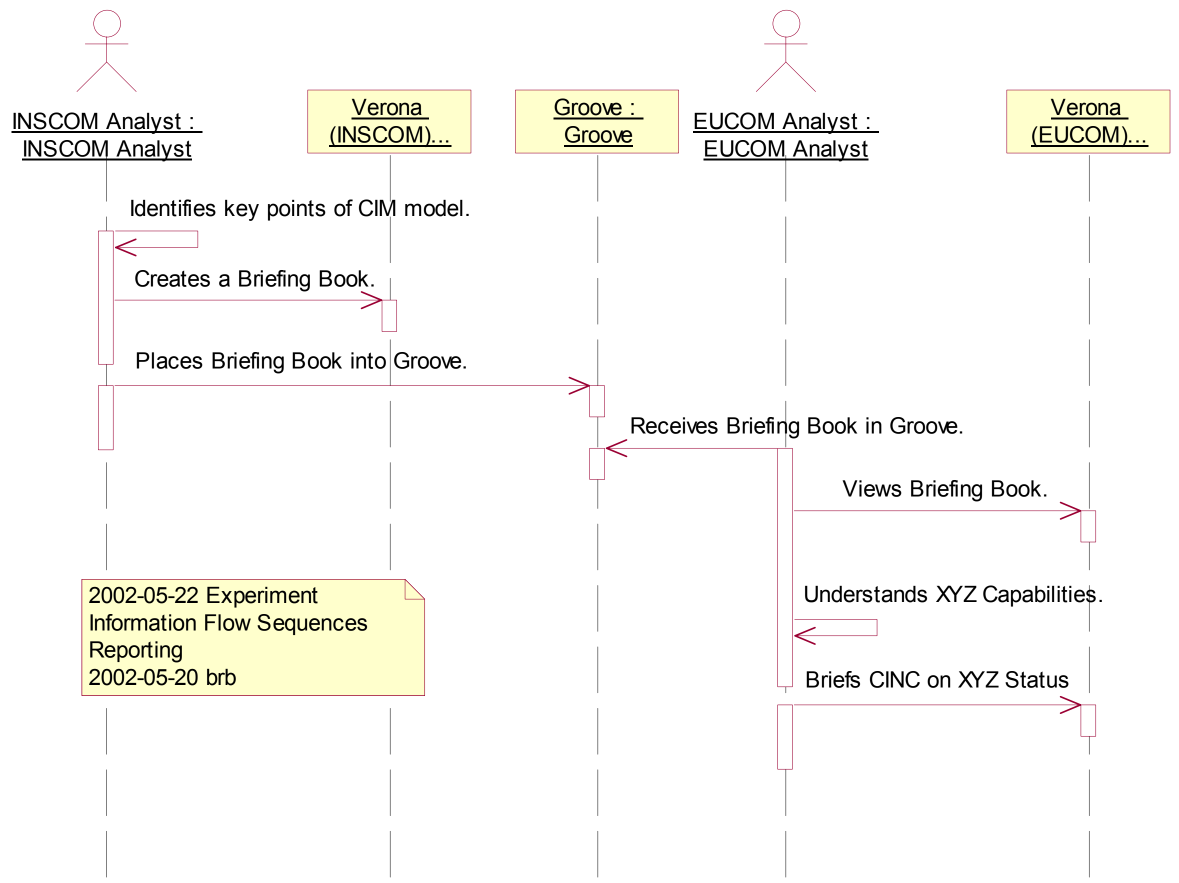
4.1.4.9 Workflow Sequences

The Workflow Sequences depict the actions occurring in the Mistral experiment across Use Cases. They are related to the collaboration diagrams showing actions and information flows in the Business Information View.

4.1.4.9.1 Matching Models and Structured Argumentation Sequence


INSCOM Situation Display : Situation Display
INSCOM Analyst : INSCOM Analyst
XMB Model Library : (XMB Model Library)
CIM : CIM
MiTap : MiTap
INSCOM Themelink : Themelink
Groove : Groove
NGIC Subject Specialist : NG...
XYZ Alert Displayed.
Watches display.
Selects a CIM Model
Opens CIM Model
Identifies gaps in CIM Model
Searches for open-source confirmation of the transfer of WMD material
Links document to the model.
Adds document to the model
2002-05-22 Experiment
Information Flow Sequence
Matching Models
Structured Argumentation
2002-05-21 brb
Opens XYZ Groove Space.
Asks for details on reweaponizing a 152mm artillery round.
Provides artillery expertise.
Receives weapons information and thinks of other concept.
Queries for concept related to NGIC Weapons expertise.
Relates Results to CIM Model.
Adds document to the model.
Shares CIM Model.

4.1.4.9.2 Reporting Sequence

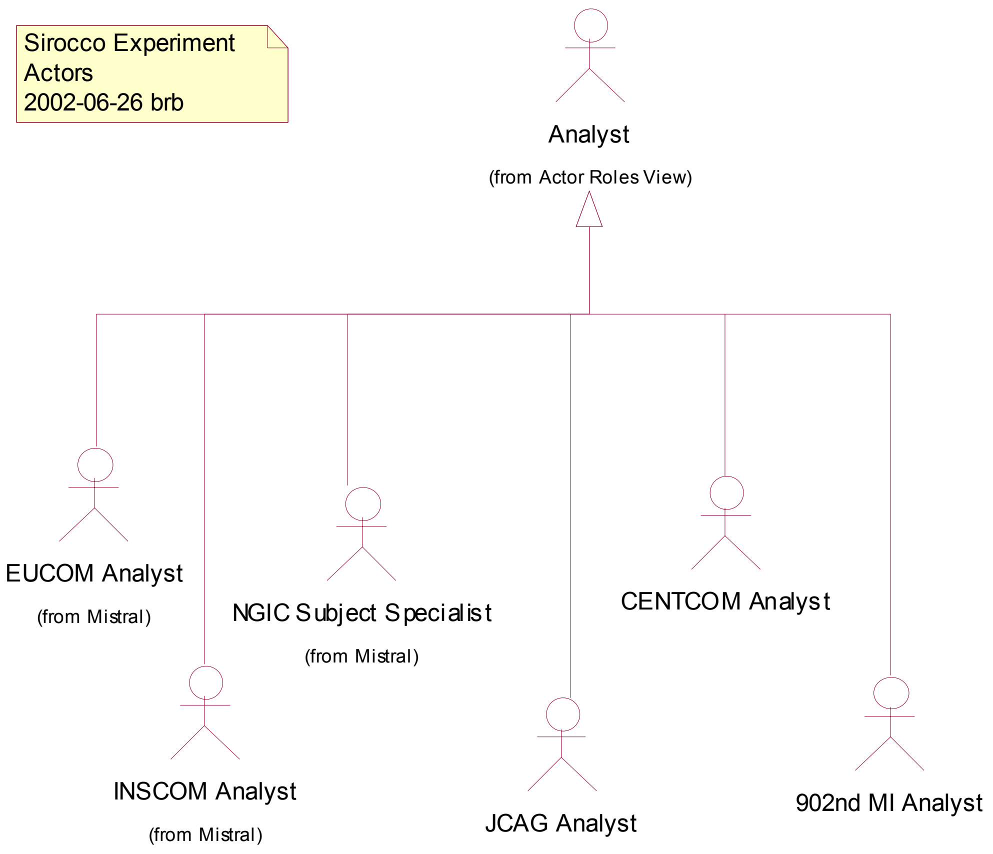

INSCOM Analyst : INSCOM Analyst
Verona (INSCOM)...
Groove : Groove
EUCOM Analyst : EUCOM Analyst
Verona (EUCOM)...
Identifies key points of CIM model.
Creates a Briefing Book.
Places Briefing Book into Groove.
Receives Briefing Book in Groove.
Views Briefing Book.
Understands XYZ Capabilities.
Briefs CINC on XYZ Status
2002-05-22 Experiment
Information Flow Sequences
Reporting
2002-05-20 brb

4.2. Sirocco Experiment (August 2002 - Planned)

4.2.1. Actor Roles View

Five Actors Roles are used in the Sirocco Experiments.

- INSCOM Analyst
- NGIC Subject Specialist
- EUCOM Analyst
- JCAG Analyst
- CENTCOM Analyst
- 902nd MI Analyst



4.2.1.1 INSCOM Analyst

4.2.1.2 EUCOM Analyst

4.2.1.3 NGIC Subject Specialist

4.2.1.4 CENTCOM Analyst

4.2.1.5 902nd MI Analyst

4.2.1.6 JCAG Analyst

4.2.2. Objectives and Results

4.2.2.1 Objectives (Metrics for Success)

4.2.2.1.1 Create a collaborative operational network including INSCOM nodes (INSCOM, NGIC, 66th 513th 902nd MI Brigades) and at least one non-INSCOM node (JCAG).

4.2.2.1.2 Explore the value of changing policies for information sharing (how to bridge CT & CI operationally, politically and technically)

4.2.2.1.3 Show automated alert visualization (concept from GIS -to- Cities of Information -to- Details).

4.2.2.1.4 Create a standard briefing-book format and at least two structured argument templates pertaining to the Sirocco problem set.

4.2.2.1.5 Include operational users early by focusing on a real-world problem leveraging existing real-world data (law-enforcement and IC).

4.2.2.1.6 Establish an infrastructure of software, hardware and people that will serve as the basis for future experiments and future IAC deployments.

4.2.2.1.7 Integrate Genoa, EELD, TIDES, and INSCOM tools and data to create end-to-end functionality.

4.2.2.2 Sources of Potentially New Insights

4.2.2.3 The operational nature and data being used in the experiment have the potential to create unique insights and contribute to solving a real-world problem.

4.2.2.4 Models of real-world problems

4.2.2.5 Metrics and milestones associated with CI and CT collaboration on an established peer-to-peer network

4.2.3. Results

4.2.4. Experiment Deployment View

The Experiment Deployment view describes the mapping(s) of the software, identified in the Business Application Service View and used in Experiments, onto the hardware and reflects its distributed aspect. It is analogous to the Physical View of the Rational 4 + 1 View of System Architecture.

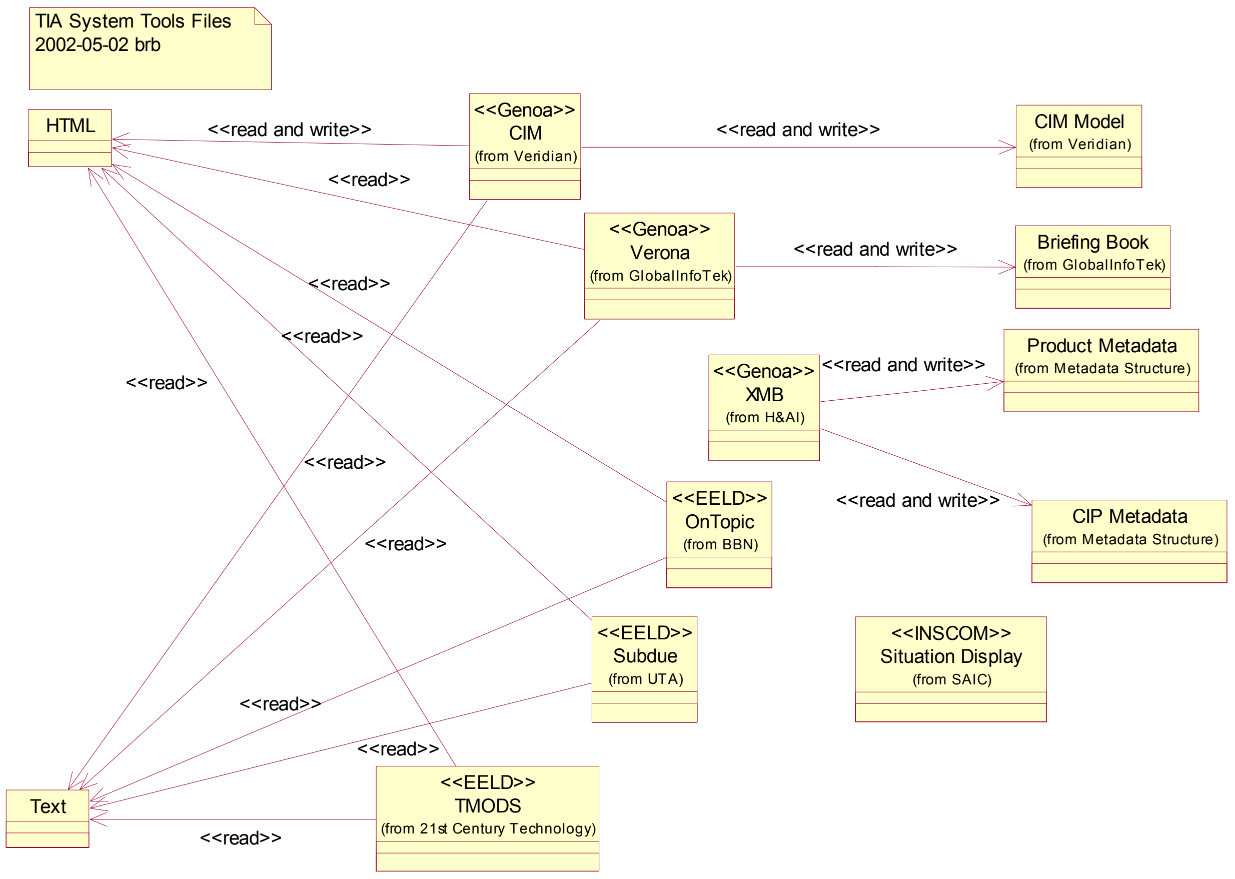
4.2.5. Business Information View

The Business Information View for the experiment describes:

- The products (formats) used in the experiment.
- The flow of roles, technology components, and information within the experiment.
- Files used by the technology components in the experiment.
- The work flows of the experiment from an information perspective.

4.2.5.1 Information Flow

4.2.5.2 Information Products



4.2.5.2.1 CIM Model

4.2.5.2.2 CIP Metadata

4.2.5.2.3 Product Metadata

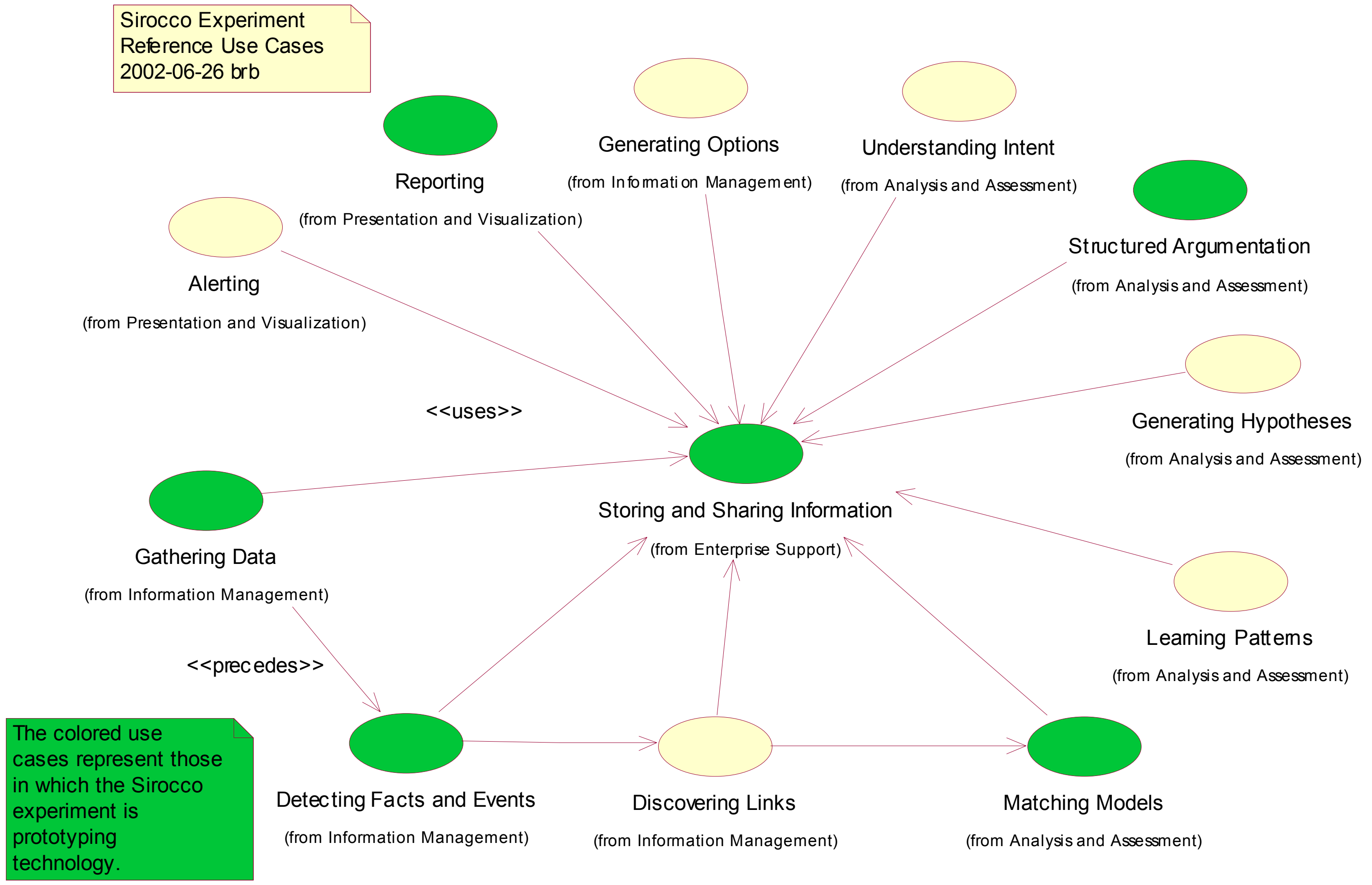
4.2.5.2.4 HTML Files

4.2.5.2.5 Text Files

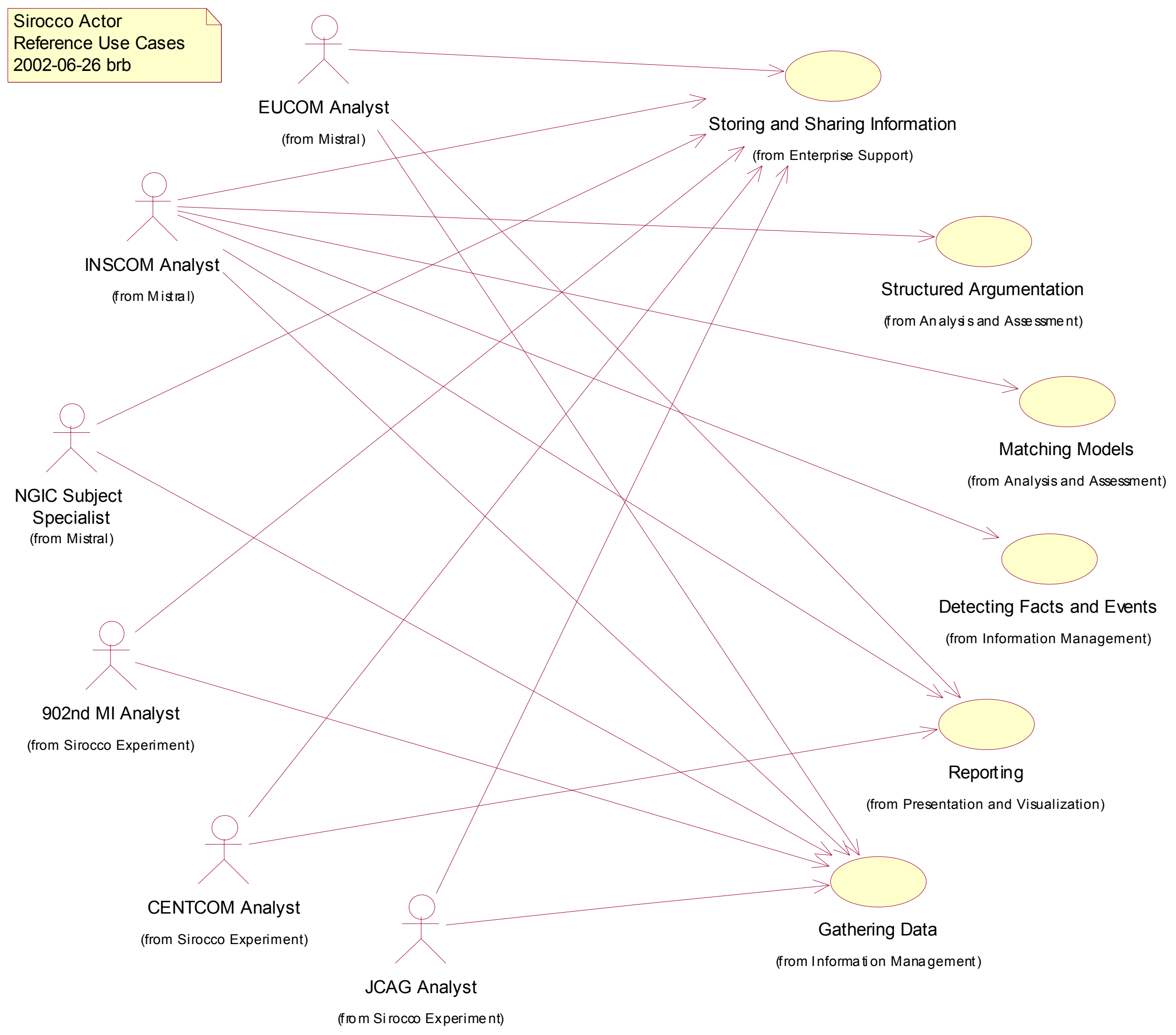
4.2.5.2.6 Briefing Book

4.2.6. Business Process View

The Sirocco Experiment includes capabilities in seven use cases of the TIA Business Model. Use Cases colored in green indicate their inclusion in the Sirocco Experiment.


Sirocco Experiment
Reference Use Cases
2002-06-26 brb
Reporting
(from Presentation and Visualization)
Generating Options
(from Information Management)
Understanding Intent
(from Analysis and Assessment)
Structured Argumentation
(from Analysis and Assessment)
Alerting
(from Presentation and Visualization)
<<uses>>
Generating Hypotheses
(from Analysis and Assessment)
Gathering Data
(from Information Management)
Storing and Sharing Information
(from Enterprise Support)
Learning Patterns
(from Analysis and Assessment)
<<precedes>>
The colored use cases represent those in which the Sirocco experiment is prototyping technology.
Detecting Facts and Events
(from Information Management)
Discovering Links
(from Information Management)
Matching Models
(from Analysis and Assessment)

4.2.6.1 Reference Use Cases


Sirocco Actor
Reference Use Cases
2002-06-26 brb
EUCOM Analyst
(from Mistral)
INSCOM Analyst
(from Mistral)
NGIC Subject Specialist
(from Mistral)
902nd MI Analyst
(from Sirocco Experiment)
CENTCOM Analyst
(from Sirocco Experiment)
JCAG Analyst
(from Sirocco Experiment)
Storing and Sharing Information
(from Enterprise Support)
Structured Argumentation
(from Analysis and Assessment)
Matching Models
(from Analysis and Assessment)
Detecting Facts and Events
(from Information Management)
Reporting
(from Presentation and Visualization)
Gathering Data
(from Information Management)

The descriptions of the Use Cases as they are used in the Sirocco scenario will be completed as the planning for the experiment continues.

4.2.6.1.1 Gathering Data

4.2.6.1.2 Detecting Facts and Events

4.2.6.1.3 Matching Models

4.2.6.1.4 Structured Argumentation

4.2.6.1.5 Reporting

4.2.6.1.6 Storing and Sharing Information

4.2.6.2 Activity Diagrams

TBD. The Activity Diagrams depict the specific interactions occurring in the Sirocco Experiment within each Use Case exercised.

4.2.6.3 Workflow Sequences

TBD. The Workflow Sequences depict the sequence of actions occurring in the Sirocco Experiment across Use Cases. The are related to the collaboration diagrams showing actions and information flows in the Business Information View.

5. Appendix A: Document Revision History

Version 1.1 – July 19, 2002	• Added Appendix B as UML for System Engineering (UML Tutorial) • Added Multi-Level Security Approach Section to the System Implementation View • Updated Business Information View to include a syntactic and semantic characterization of current products • Changed revised history to reverse-chronological and added link to revision history on title page
Version 1.0 – July 8, 2002	• Updated Information View to include Semantic Architecture • Completed Component Catalog View • Added Executive Summary • Complete initial draft of Business Application Services View
Draft Version 9 - June 27, 2002	• Integrated BAH System Model into single version • Added Initial Sirocco Experiment Model • Created Adobe PDF bookmarks for quick reference
Draft Version 8 - May 29, 2002	• Changed to multimap to accommodate separate business and system models • Inserted slides from Overview presentation
Draft Version 7 - May 23, 2002	• Modified the Actors within the system to incorporate Tom Armour's suggestions • Added icons indicating current priority areas and new content • Added the Visualizing GIS Data to the Presentation and Visualization Use Cases (Linked to Earthviewer) • Inserted bookmark hyperlinks within the model

DRAFT Version 6 – May 14, 2002	• Reduced size of IAO Logo • Incorporated feedback from Greg • Added GMF Model including CIP and Product Models • Added high-level business rules model
DRAFT Version 5 – May 9, 2002	• Updated the Actor Roles View • Created Experiment Deployment View • Changed package structure to use stereotypes to identify responsibility areas • Added experiment network topology deployment diagram
DRAFT Version 4 – May 9, 2002	• Updated diagrams for revised package architecture • Updated documentation for all • Updated canonical use case coverage map table to format properly • Modified the HTML Template • Shifted TIA Architecture Fractal Core Model below TIA Sys. Arch.
DRAFT Version 3 – May 7, 2002	• Reduced the complexity of the mindmap by controlling level of detail and hiding some branches.
DRAFT Version 2- May 7, 2002	• Added hyperlinks to diagrams to facilitate automatic PowerPoint generation. • Revised overall UML package architecture and package descriptions • Created layered System Model to be more consistent with BAH PowerPoint sketches. • Included figures from Core Services PowerPoint presentation
DRAFT Version 1 – May 1, 2001	• B. Bebee (bebeeb@saic.com), G. Mack HA&I.

6. Appendix B: UML for System Engineering

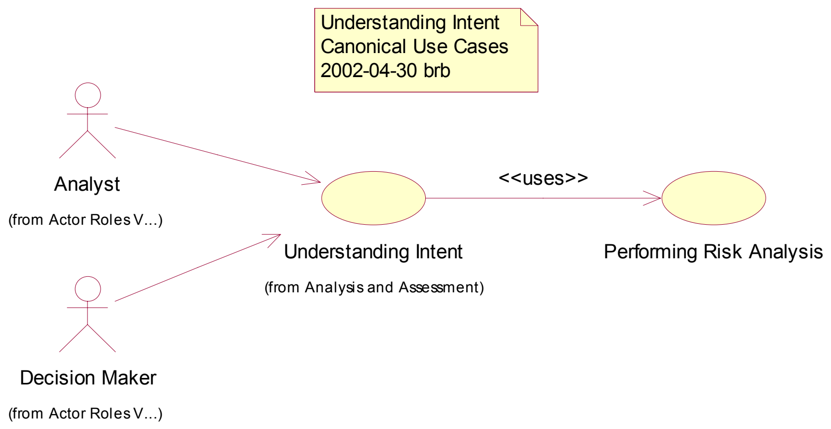
6.1. *The Unified Modeling Language as a System Engineering Aid*

- **Commonly Understood Notation**
  - **Notation most closely related to Rumbaugh's OMT.**
  - **OMG Standard**
- **Unambiguous Way of Capturing:**
  - **System Architecture**
  - **System Operations**
  - **Deployed Embodiment**
- **Containers for Indexing Attributes, Operations, & MOMs across:**
  - **Categories of System Entities**
  - **Actors**
  - **Deployed System Entities**

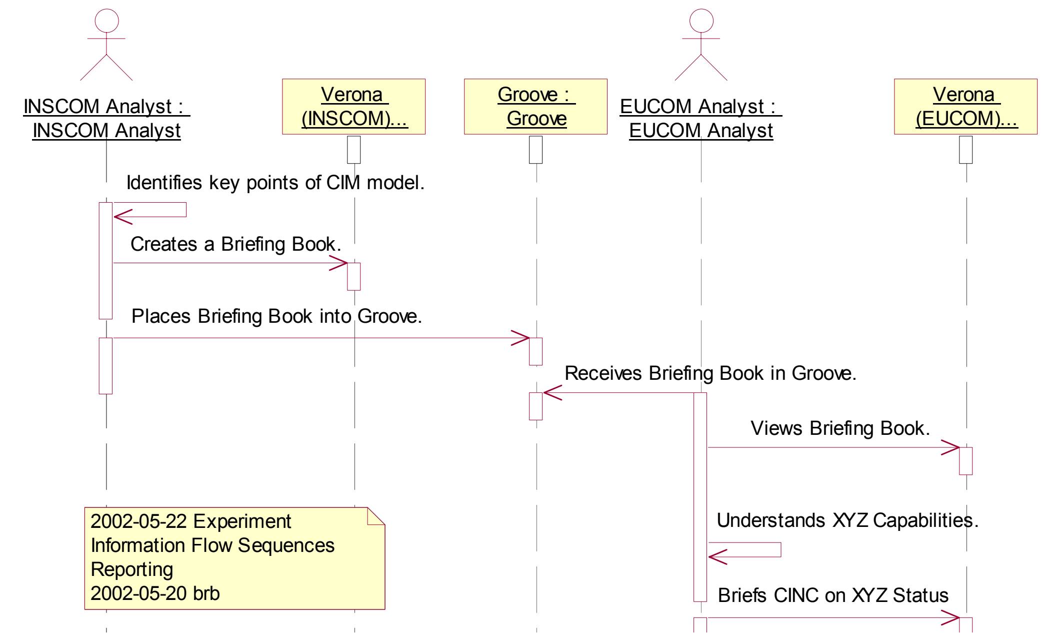
6.2. *UML as a Modeling Tool, System Operations*

- **Use Cases = Visual Index of Documented Uses**
- **Sequence Diagrams = Time-Sequenced Interactions Between System Entities**
- **Collaboration Diagrams = Inventory of Interactions Between System Entities**
- **Activity Diagrams / State Diagrams (not shown) = Internal Process Flows**

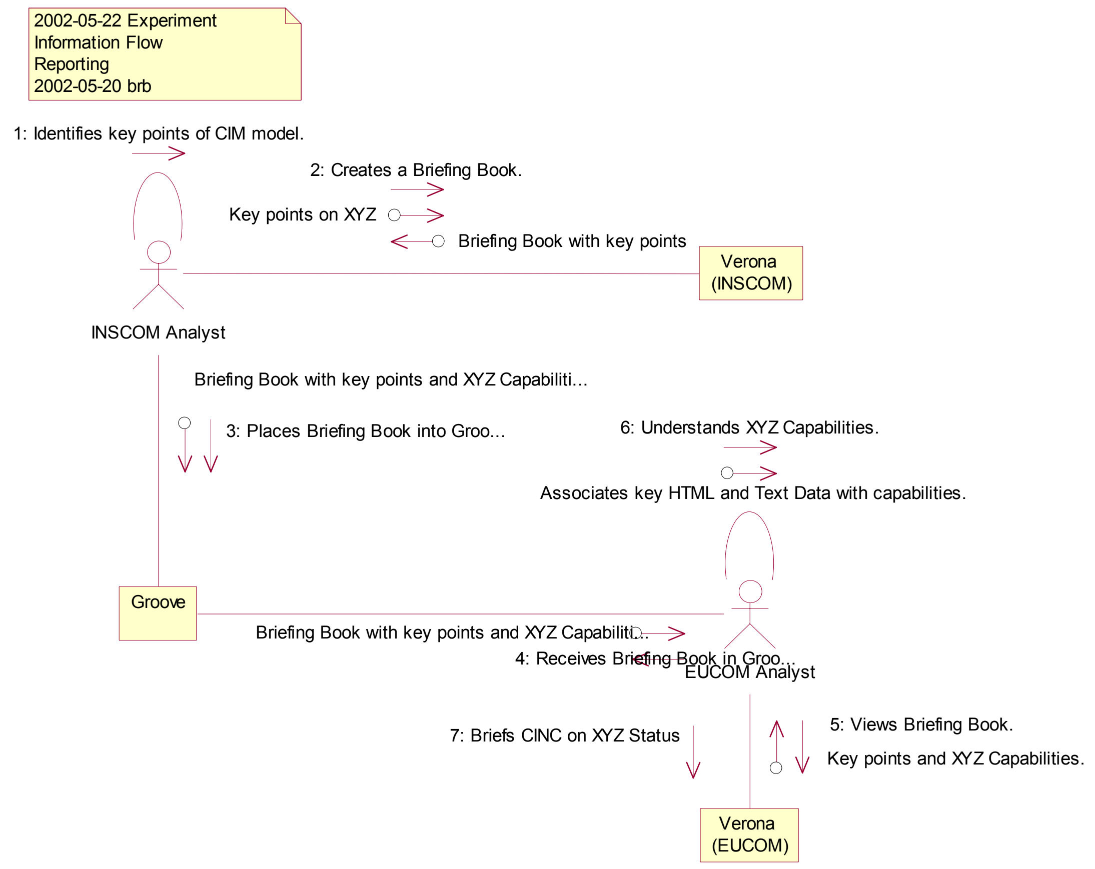
6.2.1. Use Case Diagram Example



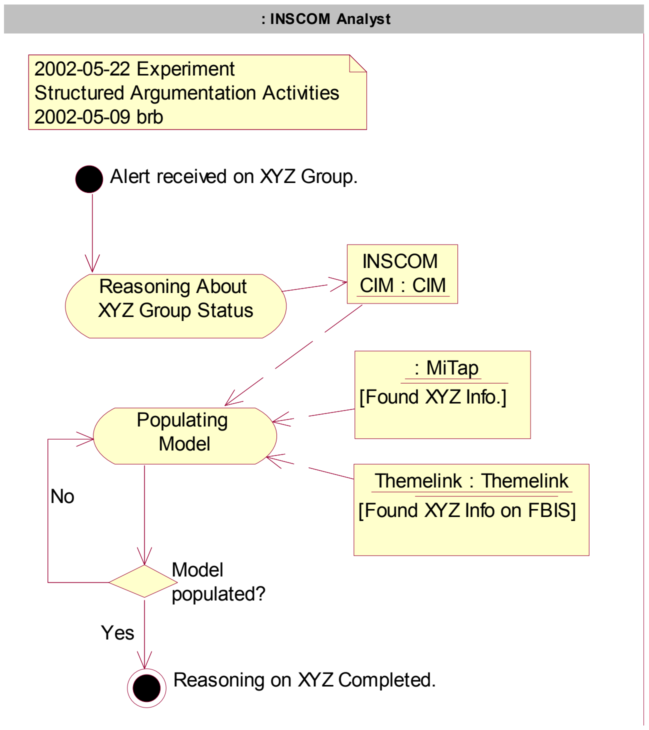
6.2.2. Sequence Diagram Example


INSCOM Analyst : INSCOM Analyst
Verona (INSCOM)...
Groove : Groove
EUCOM Analyst : EUCOM Analyst
Verona (EUCOM)...
Identifies key points of CIM model.
Creates a Briefing Book.
Places Briefing Book into Groove.
Receives Briefing Book in Groove.
Views Briefing Book.
Understands XYZ Capabilities.
Briefs CINC on XYZ Status
2002-05-22 Experiment
Information Flow Sequences
Reporting
2002-05-20 brb

6.2.3. Collaboration Diagram Example

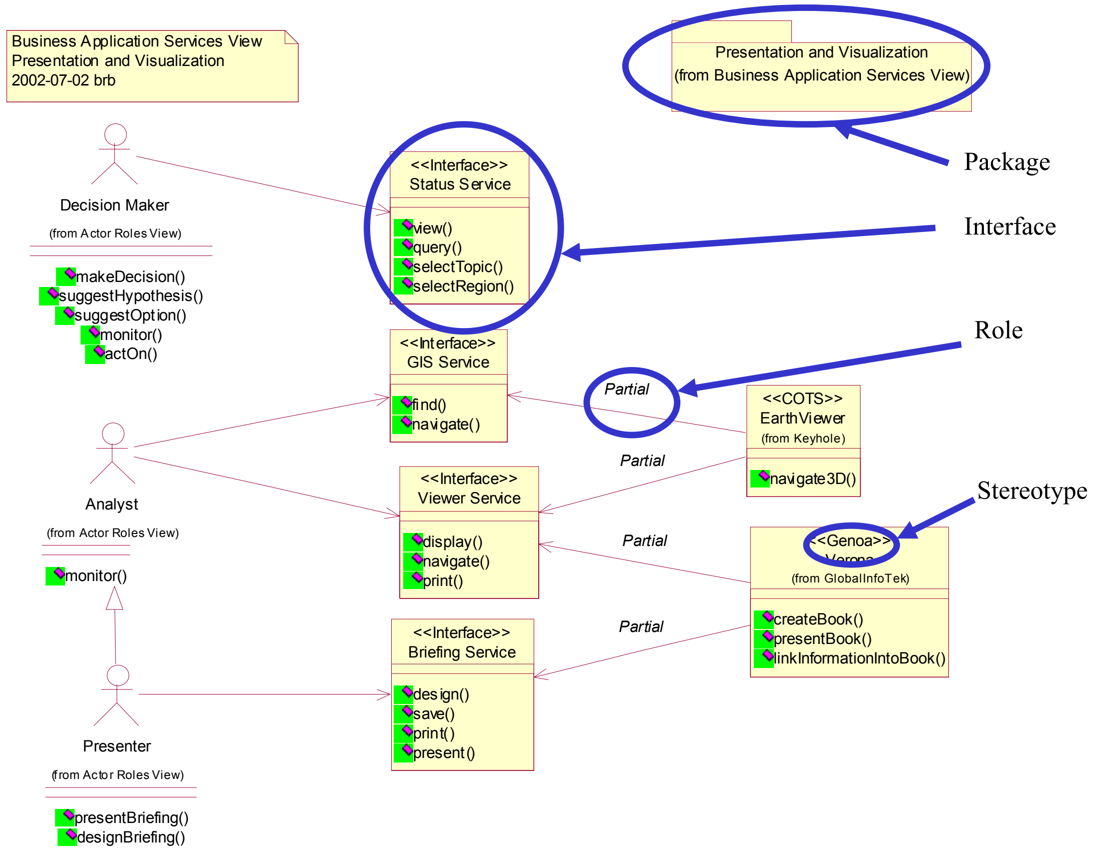

2002-05-22 Experiment
Information Flow
Reporting
2002-05-20 brb
1: Identifies key points of CIM model.
2: Creates a Briefing Book.
Key points on XYZ
Briefing Book with key points
Verona (INSCOM)
INSCOM Analyst
Briefing Book with key points and XYZ Capabiliti...
3: Places Briefing Book into Groo...
6: Understands XYZ Capabilities.
Associates key HTML and Text Data with capabilities.
Groove
Briefing Book with key points and XYZ Capabiliti...
4: Receives Briefing Book in Groo...
EUCOM Analyst
7: Briefs CINC on XYZ Status
5: Views Briefing Book.
Key points and XYZ Capabilities.
Verona (EUCOM)

6.2.4. Activity Diagram Example



6.3. UML as a Modeling Tool, Multiple Views

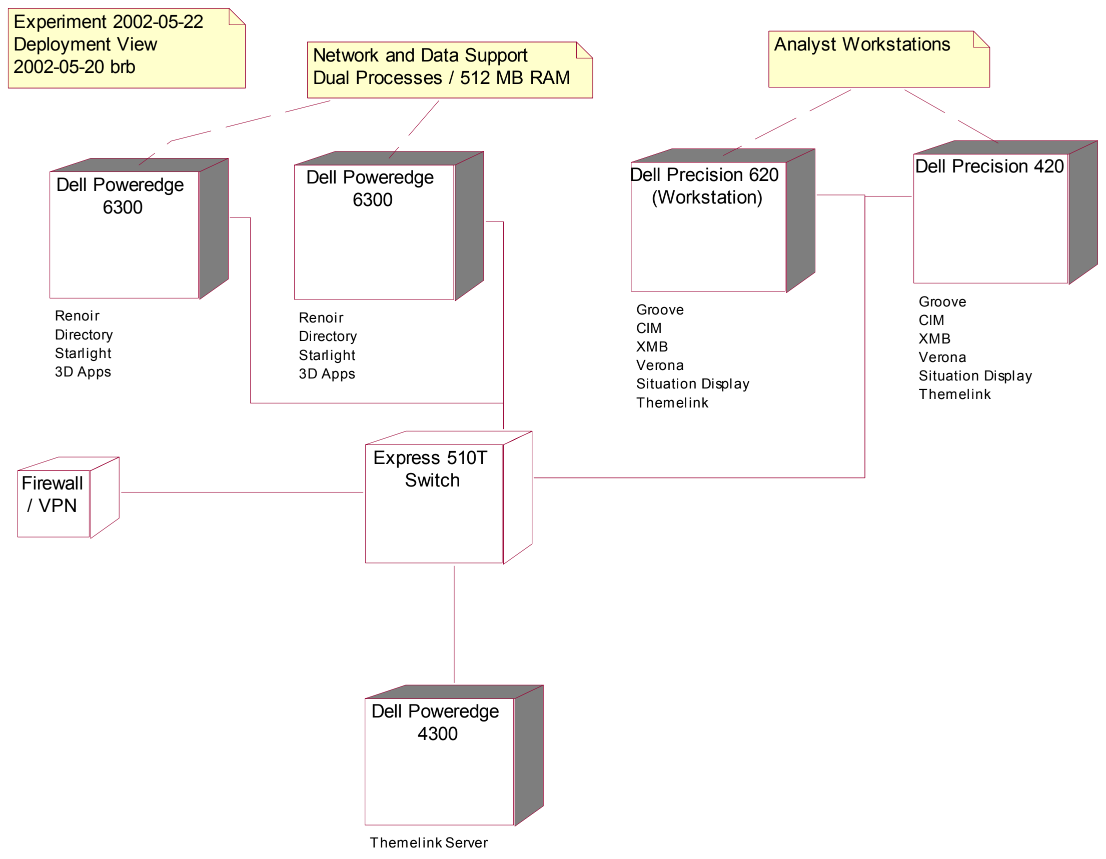
- **Packages (not shown) = "A general purpose mechanism for organizing elements into groups."**
- **Interfaces = "a type to describe the externally visible behavior of a class, object, or other entity."**
- **Roles = "named specific behavior of an entity participating in a particular context."**
- **Stereotypes = "extends the semantics of the meta-model, but not the structure " -- for both classes and associations.**



6.4. UML As Modeling Tool, System Implementation

- **Deployment Diagrams**
  - **Shows the configuration of run-time processing elements and the software components, processes, and objects that live on them.**
- **Component Diagrams (Not Shown)**
  - **shows the dependencies among software components**

6.4.1. Deployment Diagram Example



7. Appendix C Services WSDL Specifications

7.1. C.1 Security Services WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions              targetNamespace="              urn:Security"
xmlns="http://schemas.xmlsoap.org/wsdl/"                     xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"   xmlns:impl="  urn:Security-
impl"                       xmlns:intf="                     urn:Security"
xmlns:tns2="http://domain.security.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
     <schema      targetNamespace="http://domain.security.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
        <complexType name="User">
           <sequence>
              <element name="firstName" nillable="true" type="xsd:string"/>
              <element name="lastName" nillable="true" type="xsd:string"/>
              <element name="phone" nillable="true" type="xsd:string"/>
              <element name="email" nillable="true" type="xsd:string"/>
              <element name="role" nillable="true" type="tns2:Role"/>
              <element name="List" nillable="true" type="SOAP-ENC:Array"/>
           </sequence>
        </complexType>
        <complexType name="Role">
           <sequence>
              <element name="name" nillable="true" type="xsd:string"/>
           </sequence>
        </complexType>
        <element name="User" nillable="true" type="tns2:User"/>
        <element name="Role" nillable="true" type="tns2:Role"/>
     </schema>
     <schema      targetNamespace="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="Array" nillable="true" type="SOAP-ENC:Array"/>
     </schema>
  </types>
  <wsdl:message name="getUserRequest">
     <wsdl:part name="in0" type="SOAP-ENC:string"/>
  </wsdl:message>
  <wsdl:message name="EditUserRequest">
     <wsdl:part name="in0" type="tns2:User"/>
  </wsdl:message>
  <wsdl:message name="authorizeResponse"/>
  <wsdl:message name="EditUserResponse"/>
  <wsdl:message name="addUserRequest">
     <wsdl:part name="in0" type="tns2:User"/>
  </wsdl:message>
  <wsdl:message name="removeUserFromRoleRequest">
     <wsdl:part name="in0" type="tns2:User"/>
     <wsdl:part name="in1" type="tns2:Role"/>
  </wsdl:message>
  <wsdl:message name="authenticateRequest">
     <wsdl:part name="in0" type="SOAP-ENC:string"/>
     <wsdl:part name="in1" type="SOAP-ENC:string"/>
  </wsdl:message>
  <wsdl:message name="assignUserToRoleRequest">
     <wsdl:part name="in0" type="tns2:User"/>
     <wsdl:part name="in1" type="tns2:Role"/>
  </wsdl:message>
  <wsdl:message name="getUsersRequest"/>
  <wsdl:message name="RemoveUserResponse"/>
  <wsdl:message name="getUsersResponse">
     <wsdl:part name="return" type="SOAP-ENC:Array"/>
  </wsdl:message>
  <wsdl:message name="addUserResponse"/>
  <wsdl:message name="assignUserToRoleResponse"/>
  <wsdl:message name="authorizeRequest">
     <wsdl:part name="in0" type="SOAP-ENC:string"/>
     <wsdl:part name="in1" type="SOAP-ENC:string"/>
  </wsdl:message>
  <wsdl:message name="getUserResponse">
```

```
      <wsdl:part name="return" type="tns2:User"/>
   </wsdl:message>
   <wsdl:message name="SecurityManagerException"/>
   <wsdl:message name="RemoveUserRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="removeUserFromRoleResponse"/>
   <wsdl:message name="authenticateResponse"/>
   <wsdl:portType name="SecurityManager">
      <wsdl:operation name="EditUser" parameterOrder="in0">
         <wsdl:input message="intf:EditUserRequest"/>
         <wsdl:output message="intf:EditUserResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="RemoveUser" parameterOrder="in0">
         <wsdl:input message="intf:RemoveUserRequest"/>
         <wsdl:output message="intf:RemoveUserResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="addUser" parameterOrder="in0">
         <wsdl:input message="intf:addUserRequest"/>
         <wsdl:output message="intf:addUserResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="assignUserToRole" parameterOrder="in0 in1">
         <wsdl:input message="intf:assignUserToRoleRequest"/>
         <wsdl:output message="intf:assignUserToRoleResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="authenticate" parameterOrder="in0 in1">
         <wsdl:input message="intf:authenticateRequest"/>
         <wsdl:output message="intf:authenticateResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="authorize" parameterOrder="in0 in1">
         <wsdl:input message="intf:authorizeRequest"/>
         <wsdl:output message="intf:authorizeResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="getUser" parameterOrder="in0">
         <wsdl:input message="intf:getUserRequest"/>
         <wsdl:output message="intf:getUserResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="getUsers">
         <wsdl:input message="intf:getUsersRequest"/>
         <wsdl:output message="intf:getUsersResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="removeUserFromRole" parameterOrder="in0 in1">
         <wsdl:input message="intf:removeUserFromRoleRequest"/>
         <wsdl:output message="intf:removeUserFromRoleResponse"/>
         <wsdl:fault              message="intf:SecurityManagerException"
name="SecurityManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="securitySoapBinding" type="intf:SecurityManager">
      <wsdlsoap:binding                                          style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="EditUser">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="EditUser" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
```

```
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="RemoveUser">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="RemoveUser" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="addUser">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="addUser" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="assignUserToRole">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="assignUserToRole" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="authenticate">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="authenticate" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="authorize">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="authorize" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getUser">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getUser" use="encoded"/>
```

```
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getUsers">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getUsers" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="removeUserFromRole">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="removeUserFromRole" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Security" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="SecurityManagerService">
      <wsdl:port binding="intf:securitySoapBinding" name="security">
         <wsdlsoap:address
location="http://localhost:8080/services/security"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.2. C.2 Registry Services WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions              targetNamespace="               urn:Registry"
xmlns="http://schemas.xmlsoap.org/wsdl/"                        xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:impl="  urn:Registry-
impl"                      xmlns:intf="                        urn:Registry"
xmlns:tns2="http://domain.registry.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <types>
      <schema     targetNamespace="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://www.w3.org/2001/XMLSchema">
         <element name="Array" nillable="true" type="SOAP-ENC:Array"/>
      </schema>
      <schema     targetNamespace="http://domain.registry.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
         <complexType name="RegistryEntry">
            <sequence/>
         </complexType>
         <element            name="RegistryEntry"            nillable="true"
type="tns2:RegistryEntry"/>
      </schema>
   </types>
   <wsdl:message name="getServicesResponse">
      <wsdl:part name="return" type="SOAP-ENC:Array"/>
   </wsdl:message>
   <wsdl:message name="removeServiceRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="getByNameRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="editServiceRequest">
      <wsdl:part name="in0" type="tns2:RegistryEntry"/>
   </wsdl:message>
   <wsdl:message name="RegistryManagerException"/>
   <wsdl:message name="getByDepartmentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="editServiceResponse"/>
   <wsdl:message name="registerServiceResponse"/>
   <wsdl:message name="getByDepartmentResponse">
      <wsdl:part name="return" type="SOAP-ENC:Array"/>
   </wsdl:message>
   <wsdl:message name="getByNameResponse">
      <wsdl:part name="return" type="SOAP-ENC:Array"/>
   </wsdl:message>
   <wsdl:message name="removeServiceResponse"/>
   <wsdl:message name="getServicesRequest"/>
   <wsdl:message name="registerServiceRequest">
      <wsdl:part name="in0" type="tns2:RegistryEntry"/>
   </wsdl:message>
   <wsdl:portType name="RegistryManager">
      <wsdl:operation name="getByName" parameterOrder="in0">
         <wsdl:input message="intf:getByNameRequest"/>
         <wsdl:output message="intf:getByNameResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="editService" parameterOrder="in0">
         <wsdl:input message="intf:editServiceRequest"/>
         <wsdl:output message="intf:editServiceResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="getByDepartment" parameterOrder="in0">
         <wsdl:input message="intf:getByDepartmentRequest"/>
         <wsdl:output message="intf:getByDepartmentResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="getServices">
         <wsdl:input message="intf:getServicesRequest"/>
         <wsdl:output message="intf:getServicesResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="registerService" parameterOrder="in0">
         <wsdl:input message="intf:registerServiceRequest"/>
         <wsdl:output message="intf:registerServiceResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="removeService" parameterOrder="in0">
         <wsdl:input message="intf:removeServiceRequest"/>
         <wsdl:output message="intf:removeServiceResponse"/>
         <wsdl:fault              message="intf:RegistryManagerException"
name="RegistryManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="registrySoapBinding" type="intf:RegistryManager">
      <wsdlsoap:binding                                         style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getByName">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getByName" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="editService">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="editService" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getByDepartment">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getByDepartment" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getServices">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getServices" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="registerService">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="registerService" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="removeService">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="removeService" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Registry" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="RegistryManagerService">
      <wsdl:port binding="intf:registrySoapBinding" name="registry">
         <wsdlsoap:address
location="http://localhost:8080/services/registry"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.3. C.3 Data Services WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions              targetNamespace="              urn:Repository"
xmlns="http://schemas.xmlsoap.org/wsdl/"                         xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:impl="  urn:Repository-
impl"                        xmlns:intf="                     urn:Repository"
xmlns:tns2="http://domain.repository.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <types>
      <schema   targetNamespace="http://domain.repository.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
         <complexType name="Content">
            <sequence/>
         </complexType>
         <element name="Content" nillable="true" type="tns2:Content"/>
         <complexType name="Schema">
            <sequence/>
         </complexType>
         <element name="Schema" nillable="true" type="tns2:Schema"/>
      </schema>
      <schema     targetNamespace="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://www.w3.org/2001/XMLSchema">
         <element name="Array" nillable="true" type="SOAP-ENC:Array"/>
      </schema>
   </types>
   <wsdl:message name="addContentResponse">
      <wsdl:part name="return" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="removeContentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="unregisterSchemaResponse"/>
   <wsdl:message name="addContentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="tns2:Content"/>
   </wsdl:message>
   <wsdl:message name="registerSchemaResponse">
      <wsdl:part name="return" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="searchContentResponse">
      <wsdl:part name="return" type="SOAP-ENC:Array"/>
   </wsdl:message>
   <wsdl:message name="searchContentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="getContentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="updateContentRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="tns2:Content"/>
   </wsdl:message>
   <wsdl:message name="updateContentResponse"/>
   <wsdl:message name="removeContentResponse"/>
   <wsdl:message name="unregisterSchemaRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="getContentResponse"/>
   <wsdl:message name="registerSchemaRequest">
      <wsdl:part name="in0" type="tns2:Schema"/>
   </wsdl:message>
   <wsdl:message name="RepositoryManagerException"/>
   <wsdl:portType name="RepositoryManager">
      <wsdl:operation name="getContent" parameterOrder="in0 in1">
         <wsdl:input message="intf:getContentRequest"/>
         <wsdl:output message="intf:getContentResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="addContent" parameterOrder="in0 in1">
         <wsdl:input message="intf:addContentRequest"/>
         <wsdl:output message="intf:addContentResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="registerSchema" parameterOrder="in0">
         <wsdl:input message="intf:registerSchemaRequest"/>
         <wsdl:output message="intf:registerSchemaResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="removeContent" parameterOrder="in0 in1">
         <wsdl:input message="intf:removeContentRequest"/>
         <wsdl:output message="intf:removeContentResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="searchContent" parameterOrder="in0 in1">
         <wsdl:input message="intf:searchContentRequest"/>
         <wsdl:output message="intf:searchContentResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="unregisterSchema" parameterOrder="in0">
         <wsdl:input message="intf:unregisterSchemaRequest"/>
         <wsdl:output message="intf:unregisterSchemaResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="updateContent" parameterOrder="in0 in1">
         <wsdl:input message="intf:updateContentRequest"/>
         <wsdl:output message="intf:updateContentResponse"/>
         <wsdl:fault                message="intf:RepositoryManagerException"
name="RepositoryManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="repositorySoapBinding" type="intf:RepositoryManager">
      <wsdlsoap:binding                                          style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getContent">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getContent" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="addContent">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="addContent" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="registerSchema">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="registerSchema" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="removeContent">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="removeContent" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="searchContent">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="searchContent" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="unregisterSchema">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="unregisterSchema" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="updateContent">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="updateContent" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Repository" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="RepositoryManagerService">
      <wsdl:port binding="intf:repositorySoapBinding" name="repository">
         <wsdlsoap:address
location="http://localhost:8080/services/repository"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.4. C.4 Messaging Services WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions              targetNamespace="              urn:Messaging"
xmlns="http://schemas.xmlsoap.org/wsdl/"                        xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:impl="  urn:Messaging-
impl"                       xmlns:intf="                       urn:Messaging"
xmlns:tns2="http://domain.messaging.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <types>
      <schema    targetNamespace="http://domain.messaging.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
         <complexType name="Topic">
```

```
                  <sequence>
                     <element name="name" nillable="true" type="xsd:string"/>
                  </sequence>
               </complexType>
               <element name="Topic" nillable="true" type="tns2:Topic"/>
               <complexType name="Message">
                  <sequence>
                     <element name="message" nillable="true" type="xsd:string"/>
                  </sequence>
               </complexType>
               <element name="Message" nillable="true" type="tns2:Message"/>
            </schema>
            <schema     targetNamespace="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://www.w3.org/2001/XMLSchema">
               <element name="Array" nillable="true" type="SOAP-ENC:Array"/>
            </schema>
         </types>
         <wsdl:message name="getMessagesRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="changeTopicPropertiesRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="subscribeToTopicResponse"/>
         <wsdl:message name="getMessagesResponse">
            <wsdl:part name="return" type="SOAP-ENC:Array"/>
         </wsdl:message>
         <wsdl:message name="MessagingManagerException"/>
         <wsdl:message name="subscribeToTopicRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="removeTopicResponse"/>
         <wsdl:message name="removeTopicRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="publishToTopicResponse"/>
         <wsdl:message name="findTopicRequest">
            <wsdl:part name="in0" type="SOAP-ENC:string"/>
         </wsdl:message>
         <wsdl:message name="createTopicResponse"/>
         <wsdl:message name="createTopicRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="changeTopicPropertiesResponse"/>
         <wsdl:message name="findTopicResponse">
            <wsdl:part name="return" type="tns2:Topic"/>
         </wsdl:message>
         <wsdl:message name="publishToTopicRequest">
            <wsdl:part name="in0" type="tns2:Topic"/>
            <wsdl:part name="in1" type="tns2:Message"/>
         </wsdl:message>
         <wsdl:portType name="MessagingManager">
            <wsdl:operation name="changeTopicProperties" parameterOrder="in0">
               <wsdl:input message="intf:changeTopicPropertiesRequest"/>
               <wsdl:output message="intf:changeTopicPropertiesResponse"/>
               <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
            </wsdl:operation>
            <wsdl:operation name="createTopic" parameterOrder="in0">
               <wsdl:input message="intf:createTopicRequest"/>
               <wsdl:output message="intf:createTopicResponse"/>
               <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
            </wsdl:operation>
            <wsdl:operation name="findTopic" parameterOrder="in0">
               <wsdl:input message="intf:findTopicRequest"/>
               <wsdl:output message="intf:findTopicResponse"/>
               <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
            </wsdl:operation>
            <wsdl:operation name="getMessages" parameterOrder="in0">
               <wsdl:input message="intf:getMessagesRequest"/>
               <wsdl:output message="intf:getMessagesResponse"/>
               <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
            </wsdl:operation>
```

```
      <wsdl:operation name="publishToTopic" parameterOrder="in0 in1">
         <wsdl:input message="intf:publishToTopicRequest"/>
         <wsdl:output message="intf:publishToTopicResponse"/>
         <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="removeTopic" parameterOrder="in0">
         <wsdl:input message="intf:removeTopicRequest"/>
         <wsdl:output message="intf:removeTopicResponse"/>
         <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="subscribeToTopic" parameterOrder="in0">
         <wsdl:input message="intf:subscribeToTopicRequest"/>
         <wsdl:output message="intf:subscribeToTopicResponse"/>
         <wsdl:fault                 message="intf:MessagingManagerException"
name="MessagingManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="messagingSoapBinding" type="intf:MessagingManager">
      <wsdlsoap:binding                                             style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="changeTopicProperties">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="changeTopicProperties" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="createTopic">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="createTopic" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="findTopic">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="findTopic" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getMessages">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getMessages" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="publishToTopic">
         <wsdlsoap:operation soapAction=""/>
```

```
        <wsdl:input>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="publishToTopic" use="encoded"/>
        </wsdl:input>
        <wsdl:output>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
        </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="removeTopic">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="removeTopic" use="encoded"/>
        </wsdl:input>
        <wsdl:output>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
        </wsdl:output>
     </wsdl:operation>
     <wsdl:operation name="subscribeToTopic">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="subscribeToTopic" use="encoded"/>
        </wsdl:input>
        <wsdl:output>
           <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Messaging" use="encoded"/>
        </wsdl:output>
     </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MessagingManagerService">
     <wsdl:port binding="intf:messagingSoapBinding" name="messaging">
        <wsdlsoap:address
location="http://localhost:8080/services/messaging"/>
     </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

7.5. C.5 Transformation Services WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions           targetNamespace="           urn:Transformation"
xmlns="http://schemas.xmlsoap.org/wsdl/"                          xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"                xmlns:impl="
urn:Transformation-impl"           xmlns:intf="           urn:Transformation"
xmlns:tns2="http://domain.transformation.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
     <schema     targetNamespace="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="Array" nillable="true" type="SOAP-ENC:Array"/>
     </schema>
     <schema
targetNamespace="http://domain.transformation.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
        <complexType name="Rule">
           <sequence/>
        </complexType>
        <element name="Rule" nillable="true" type="tns2:Rule"/>
     </schema>
  </types>
  <wsdl:message name="editRuleRequest">
     <wsdl:part name="in0" type="tns2:Rule"/>
  </wsdl:message>
  <wsdl:message name="transformResponse">
     <wsdl:part name="return" type="SOAP-ENC:string"/>
```

```
   </wsdl:message>
   <wsdl:message name="removeRuleResponse"/>
   <wsdl:message name="TransformationManagerException"/>
   <wsdl:message name="getRulesRequest"/>
   <wsdl:message name="getRuleResponse">
      <wsdl:part name="return" type="tns2:Rule"/>
   </wsdl:message>
   <wsdl:message name="removeRuleRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="addRuleRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="editRuleResponse"/>
   <wsdl:message name="getRuleRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="addRuleResponse">
      <wsdl:part name="return" type="tns2:Rule"/>
   </wsdl:message>
   <wsdl:message name="transformRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
      <wsdl:part name="in1" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="getRulesResponse">
      <wsdl:part name="return" type="SOAP-ENC:Array"/>
   </wsdl:message>
   <wsdl:portType name="TransformationManager">
      <wsdl:operation name="getRules">
         <wsdl:input message="intf:getRulesRequest"/>
         <wsdl:output message="intf:getRulesResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="addRule" parameterOrder="in0 in1">
         <wsdl:input message="intf:addRuleRequest"/>
         <wsdl:output message="intf:addRuleResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="editRule" parameterOrder="in0">
         <wsdl:input message="intf:editRuleRequest"/>
         <wsdl:output message="intf:editRuleResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="getRule" parameterOrder="in0">
         <wsdl:input message="intf:getRuleRequest"/>
         <wsdl:output message="intf:getRuleResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="removeRule" parameterOrder="in0">
         <wsdl:input message="intf:removeRuleRequest"/>
         <wsdl:output message="intf:removeRuleResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="transform" parameterOrder="in0 in1">
         <wsdl:input message="intf:transformRequest"/>
         <wsdl:output message="intf:transformResponse"/>
         <wsdl:fault              message="intf:TransformationManagerException"
name="TransformationManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding                            name="transformationSoapBinding"
type="intf:TransformationManager">
      <wsdlsoap:binding                                          style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getRules">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getRules" use="encoded"/>
```

```
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="addRule">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="addRule" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="editRule">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="editRule" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getRule">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getRule" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="removeRule">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="removeRule" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="transform">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="transform" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:Transformation" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="TransformationManagerService">
      <wsdl:port                    binding="intf:transformationSoapBinding"
name="transformation">
```

```
         <wsdlsoap:address
location="http://localhost:8080/services/transformation"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

7.6. C.6 Computational Services WSDL Service Description

```
<!-- job_manager_port_type.wsdl -->

<definitions name="JobManagerDefinition"

targetNamespace="http://jobmanager.base.ogsa.globus.org/job_manager_port_typ
e"

xmlns:tns="http://jobmanager.base.ogsa.globus.org/job_manager_port_type"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<types>
   <xsd:schema
targetNamespace="http://jobmanager.base.ogsa.globus.org/job_manager_port_typ
e"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="submitJob">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="rsl" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="submitJobResponse">
      <xsd:complexType/>
    </xsd:element>
    <xsd:element name="getStatus">
      <xsd:complexType/>
    </xsd:element>
    <xsd:element name="getStatusResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="status" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="cancel">
      <xsd:complexType/>
    </xsd:element>
    <xsd:element name="cancelResponse">
      <xsd:complexType/>
    </xsd:element>
  </xsd:schema>
</types>

<message name="SubmitJobInputMessage">
 <part name="parameters" element="tns:submitJob"/>
</message>
<message name="SubmitJobOutputMessage">
 <part name="parameters" element="tns:submitJobResponse"/>
</message>
<message name="GetStatusInputMessage">
 <part name="parameters" element="tns:getStatus"/>
</message>
<message name="GetStatusOutputMessage">
 <part name="parameters" element="tns:getStatusResponse"/>
</message>

<portType name="JobManagerPortType">
  <operation name="submitJob">
    <input message="tns:SubmitJobInputMessage"/>
    <output message="tns:SubmitJobOutputMessage"/>
  </operation>
  <operation name="getStatus">
    <input message="tns:GetStatusInputMessage"/>
    <output message="tns:GetStatusOutputMessage"/>
  </operation>
  <operation name="cancel">
```

```
    <input message="tns:CancelInputMessage"/>
    <output message="tns:CancelOutputMessage"/>
  </operation>
</portType>

</definitions>

<!-- job_manager_bindings.wsdl -->

<definitions name="JobManagerDefinition"

targetNamespace="http://jobmanager.base.ogsa.globus.org/job_manager_bindings
"
             xmlns:job-manager-port-
type="http://jobmanager.base.ogsa.globus.org/job_manager_port_type"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<import location="job_manager_port_type.wsdl"

namespace="http://jobmanager.base.ogsa.globus.org/job_manager_port_type"/>

<binding         name="JobManagerSOAPBinding"        type="job-manager-port-
type:JobManagerPortType">
  <soap:binding                                           style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="submitJob">
    <soap:operation
soapAction="http://jobmanager.base.ogsa.globus.org/job_manager#submitJob"/>
    <input>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </input>
    <output>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </output>
  </operation>
  <operation name="getStatus">
    <soap:operation
soapAction="http://jobmanager.base.ogsa.globus.org/job_manager#getStatus"/>
    <input>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </input>
    <output>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </output>
  </operation>
  <operation name="cancel">
    <soap:operation
soapAction="http://jobmanager.base.ogsa.globus.org/job_manager#cancel"/>
    <input>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </input>
    <output>
      <soap:body                                              use="literal"
namespace="http://jobmanager.base.ogsa.globus.org/job_manager"/>
    </output>
  </operation>
</binding>

</definitions>

<!-- jobmanager_service.wsdl -->

<definitions name="JobManagerDefinition"
  targetNamespace="http://jobmanager.base.ogsa.globus.org/job_manager"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:job-manager-
bindings="http://jobmanager.base.ogsa.globus.org/job_manager_bindings"
  xmlns:grid-service-
bindings="http://ogsa.gridforum.org/service/grid_service_bindings"
```

```
  xmlns:notification-source-
bindings="http://ogsa.gridforum.org/notification/notification_source_binding
s"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<import location="job_manager_bindings.wsdl"

namespace="http://jobmanager.base.ogsa.globus.org/job_manager_bindings"/>

<import location="../../core/service/grid_service_bindings.wsdl"

namespace="http://ogsa.gridforum.org/service/grid_service_bindings"/>

<import location="../../core/notification/notification_source_bindings.wsdl"

namespace="http://ogsa.gridforum.org/notification/notification_source_bindin
gs"/>

<service name="JobManagerService">
  <documentation>Globus   JobManager   service   (management   of   remote   job
execution)</documentation>

  <port binding="job-manager-bindings:JobManagerSOAPBinding"
        name="JobManagerPort">
    <soap:address location="httpg://localhost:8443/ogsa/services"/>
  </port>

  <port binding="grid-service-bindings:GridServiceSOAPBinding"
        name="GridServicePort">
    <soap:address location="httpg://localhost:8443/ogsa/services"/>
  </port>


  <port binding="notification-source-bindings:NotificationSourceSOAPBinding"
        name="NotificationSourcePort">
    <soap:address location="httpg://localhost:8080/ogsa/services"/>
  </port>


</service>

</definitions>

<!-- registry_port_type.wsdl -->

<definitions name="RegistryDefinition"

targetNamespace="http://ogsa.gridforum.org/registry/registry_port_type"

xmlns:tns="http://ogsa.gridforum.org/registry/registry_port_type"
             xmlns:ogsa-faults="http://ogsa.gridforum.org/faults"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../core/types/types.wsdl"
        namespace="http://ogsa.gridforum.org/types"/>
<import location="../../core/faults/faults.wsdl"
        namespace="http://ogsa.gridforum.org/faults"/>

<types>
  <xsd:schema
targetNamespace="http://ogsa.gridforum.org/registry/registry_port_type"
              xmlns:ogsa-types="http://ogsa.gridforum.org/types"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element              name="registerService"              type="ogsa-
types:ServiceRegistrationElementType"/>
    <xsd:element name="registerServiceResponse">
      <xsd:complexType/>
    </xsd:element>
    <xsd:element             name="unregisterService"             type="ogsa-
types:HandleElementType"/>
    <xsd:element name="unregisterServiceResponse">
      <xsd:complexType/>
    </xsd:element>
  </xsd:schema>
</types>
```

```
<message name="RegisterServiceInputMessage">
  <part name="parameters" element="tns:registerService"/>
</message>
<message name="RegisterServiceOutputMessage">
  <part name="parameters" element="tns:registerServiceResponse"/>
</message>
<message name="UnregisterServiceInputMessage">
  <part name="parameters" element="tns:unregisterService"/>
</message>
<message name="UnregisterServiceOutputMessage">
  <part name="parameters" element="tns:unregisterServiceResponse"/>
</message>

<portType name="RegistryPortType">
  <operation name="registerService">
    <input message="tns:RegisterServiceInputMessage"/>
    <output message="tns:RegisterServiceOutputMessage"/>
  </operation>
  <operation name="unregisterService">
    <input message="tns:UnregisterServiceInputMessage"/>
    <output message="tns:UnregisterServiceOutputMessage"/>
    <fault             name="HandleNotFoundFault"             message="ogsa-
faults:HandleNotFoundFault"/>
  </operation>
</portType>

</definitions>

<!-- registry_bindings.wsdl -->

<definitions name="RegistryDefinition"

targetNamespace="http://ogsa.gridforum.org/registry/registry_bindings"
             xmlns:registry-port-
type="http://ogsa.gridforum.org/registry/registry_port_type"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<import location="../../core/registry/registry_port_type.wsdl"
        namespace="http://ogsa.gridforum.org/registry/registry_port_type"/>

<binding           name="RegistrySOAPBinding"           type="registry-port-
type:RegistryPortType">
  <soap:binding                                            style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="registerService">
    <soap:operation
soapAction="http://ogsa.gridforum.org/registry/registry#registerService"/>
    <input>
      <soap:body                                              use="literal"
namespace="http://ogsa.gridforum.org/registry/registry"/>
    </input>
    <output>
      <soap:body                                              use="literal"
namespace="http://ogsa.gridforum.org/registry/registry"/>
    </output>
  </operation>
  <operation name="unregisterService">
    <soap:operation
soapAction="http://ogsa.gridforum.org/registry/registry#unregisterService"/>
    <input>
      <soap:body                                              use="literal"
namespace="http://ogsa.gridforum.org/registry/registry"/>
    </input>
    <output>
      <soap:body                                              use="literal"
namespace="http://ogsa.gridforum.org/registry/registry"/>
    </output>
    <fault>
      <soap:fault name="HandleNotFoundFault" use="encoded" style="rpc"/>
    </fault>
  </operation>
</binding>

</definitions>
```

```
<!-- vo_registry_service.wsdl -->

<definitions name="VORegistryDefinition"
  targetNamespace="http://registry.base.ogsa.globus.org/vo_registry"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:registry-inspection-
bindings="http://ogsa.gridforum.org/registry/registry_inspection_bindings"
  xmlns:registry-
bindings="http://ogsa.gridforum.org/registry/registry_bindings"
  xmlns:grid-service-
bindings="http://ogsa.gridforum.org/service/grid_service_bindings"
  xmlns:notification-source-
bindings="http://ogsa.gridforum.org/notification/notification_source_binding
s"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<import location="../../core/registry/registry_inspection_bindings.wsdl"

namespace="http://ogsa.gridforum.org/registry/registry_inspection_bindings"/
>

<import location="../../core/registry/registry_bindings.wsdl"
        namespace="http://ogsa.gridforum.org/registry/registry_bindings"/>

<import location="../../core/service/grid_service_bindings.wsdl"

namespace="http://ogsa.gridforum.org/service/grid_service_bindings"/>

<import location="../../core/notification/notification_source_bindings.wsdl"

namespace="http://ogsa.gridforum.org/notification/notification_source_bindin
gs"/>

<service name="VORegistryService">
  <documentation>persistent service managing a registry of services in a
virtual organization</documentation>

  <port binding="registry-inspection-bindings:RegistryInspectionSOAPBinding"
        name="RegistryInspectionPort">
    <soap:address location="http://localhost:8080/ogsa/services"/>
  </port>

  <port binding="registry-bindings:RegistrySOAPBinding"
        name="RegistryPort">
    <soap:address location="http://localhost:8080/ogsa/services"/>
  </port>

  <port binding="grid-service-bindings:GridServiceSOAPBinding"
        name="GridServicePort">
    <soap:address location="http://localhost:8080/ogsa/services"/>
  </port>

  <port binding="notification-source-bindings:NotificationSourceSOAPBinding"
        name="NotificationSourcePort">
    <soap:address location="http://localhost:8080/ogsa/services"/>
  </port>
</service>

</definitions>
```

7.7. C.7 Services Management WSDL Service Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions          targetNamespace="          urn:ServicesManagement"
xmlns="http://schemas.xmlsoap.org/wsdl/"                           xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"                  xmlns:impl="
urn:ServicesManagement-impl"       xmlns:intf="       urn:ServicesManagement"
xmlns:tns2="http://domain.servicesmanagement.core.tia.darpa.mil"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <types>
      <schema
targetNamespace="http://domain.servicesmanagement.core.tia.darpa.mil"
xmlns="http://www.w3.org/2001/XMLSchema">
         <complexType name="Topic">
```

```
            <sequence>
               <element name="name" nillable="true" type="xsd:string"/>
            </sequence>
         </complexType>
         <element name="Topic" nillable="true" type="tns2:Topic"/>
      </schema>
   </types>
   <wsdl:message name="createServicesTopicRequest">
      <wsdl:part name="in0" type="tns2:Topic"/>
   </wsdl:message>
   <wsdl:message name="publishToServiceTopicResponse"/>
   <wsdl:message name="ServicesManagementManagerException"/>
   <wsdl:message name="createServicesTopicResponse"/>
   <wsdl:message name="subscribeToServiceTopicRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:message name="subscribeToServiceTopicResponse"/>
   <wsdl:message name="publishToServiceTopicRequest">
      <wsdl:part name="in0" type="SOAP-ENC:string"/>
   </wsdl:message>
   <wsdl:portType name="ServicesManagementManager">
      <wsdl:operation name="createServicesTopic" parameterOrder="in0">
         <wsdl:input message="intf:createServicesTopicRequest"/>
         <wsdl:output message="intf:createServicesTopicResponse"/>
         <wsdl:fault        message="intf:ServicesManagementManagerException"
name="ServicesManagementManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="publishToServiceTopic" parameterOrder="in0">
         <wsdl:input message="intf:publishToServiceTopicRequest"/>
         <wsdl:output message="intf:publishToServiceTopicResponse"/>
         <wsdl:fault        message="intf:ServicesManagementManagerException"
name="ServicesManagementManagerException"/>
      </wsdl:operation>
      <wsdl:operation name="subscribeToServiceTopic" parameterOrder="in0">
         <wsdl:input message="intf:subscribeToServiceTopicRequest"/>
         <wsdl:output message="intf:subscribeToServiceTopicResponse"/>
         <wsdl:fault        message="intf:ServicesManagementManagerException"
name="ServicesManagementManagerException"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding                          name="servicesmanagementSoapBinding"
type="intf:ServicesManagementManager">
      <wsdlsoap:binding                                                style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="createServicesTopic">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="createServicesTopic" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:ServicesManagement" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="publishToServiceTopic">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="publishToServiceTopic" use="encoded"/>
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:ServicesManagement" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="subscribeToServiceTopic">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="subscribeToServiceTopic" use="encoded"/>
```

```
         </wsdl:input>
         <wsdl:output>
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"         namespace="
urn:ServicesManagement" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="ServicesManagementManagerService">
      <wsdl:port                 binding="intf:servicesmanagementSoapBinding"
name="servicesmanagement">
         <wsdlsoap:address
location="http://localhost:8080/services/servicesmanagement"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

8. Appendix D: System Design Viewer

The System Design Viewer will provide a mechanism for effectively navigating the System Description, which is currently under development. A prototype demonstration is expected to take place in early July, 2002.

8.1. End User Requirements

8.1.1. Read

8.1.1.1 External Documentation Linked to Model Entities

8.1.2. Display

8.1.2.1 Display UML Diagrams

8.1.2.2 Display the Model Entities

8.1.3. Navigate

8.1.3.1 Navigate through the Architecture Using Diagram Elements

8.1.3.2 Navigate through the Architecture Using Entity Associations

8.1.4. Search and Discovery

8.1.4.1 Model Entity Properties

8.1.4.2 External Documentation

8.1.5. Add

8.1.5.1 External Documentation to Model Entities

8.1.5.2 Annotations to Model Entities

8.1.6. Modify

8.1.6.1 Merge XMI from Business and System Models

8.1.6.2 Update Model Entity Metadata

8.1.7. Delete

8.1.7.1 Mark Model Entities for Deletion in Rational Rose

8.1.8. Print

8.1.8.1 All Model Entities

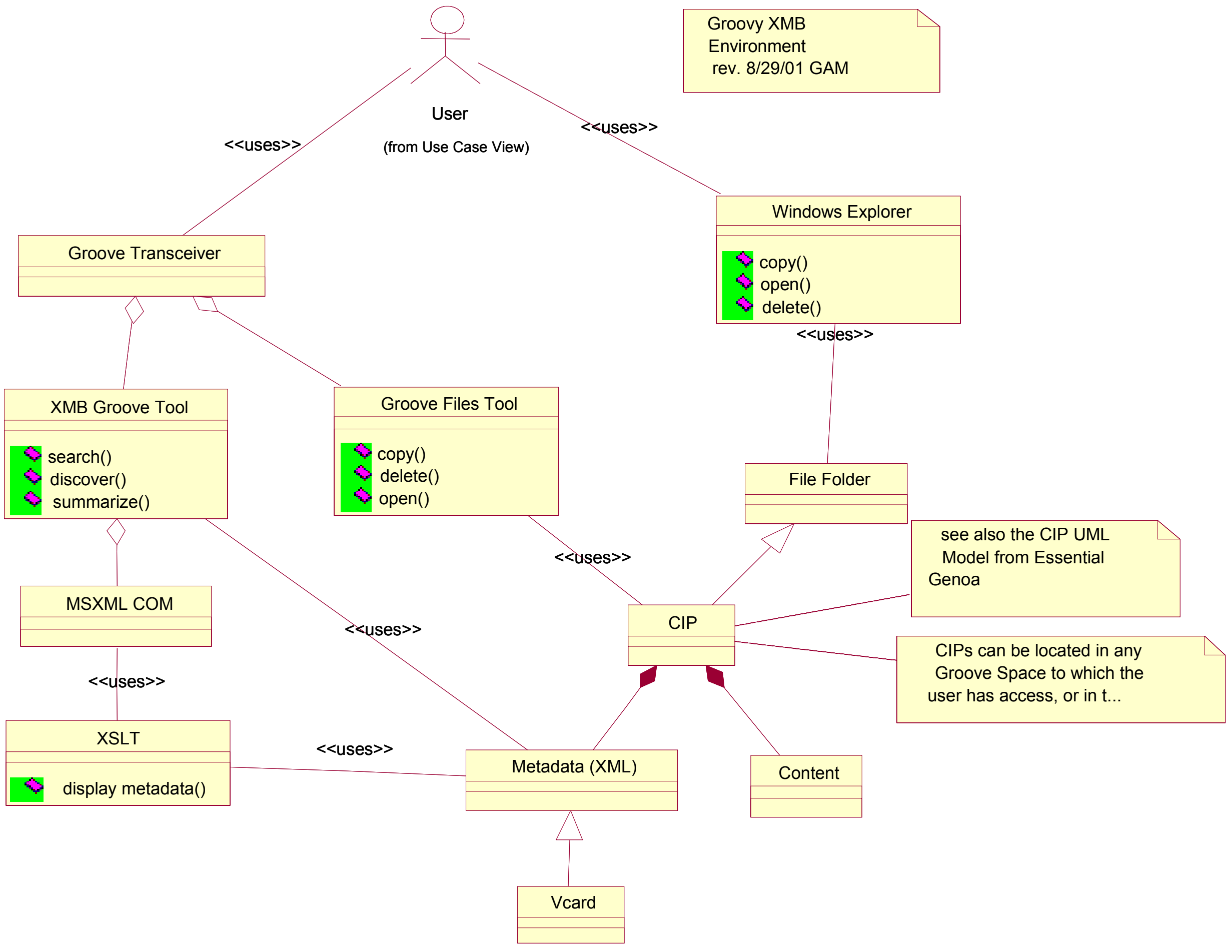
8.1.8.2 Selected Model Entities

8.1.8.3 Report of Search Results Including Model Entities, Diagrams, External Documentation

8.1.8.4 Selected Diagrams

8.2. Design

The XMB is a JavaScript/MSXML COM/XSLT application running as a tool in a Groove environment using the Groove APIs.


Groovy XMB Environment rev. 8/29/01 GAM
User
(from Use Case View)
<<uses>>
<<uses>>
Windows Explorer
copy()
open()
delete()
Groove Transceiver
<<uses>>
XMB Groove Tool
search()
discover()
summarize()
Groove Files Tool
copy()
delete()
open()
File Folder
see also the CIP UML Model from Essential Genoa
<<uses>>
MSXML COM
<<uses>>
CIP
CIPs can be located in any Groove Space to which the user has access, or in t...
<<uses>>
XSLT
display metadata()
<<uses>>
Metadata (XML)
Content
Vcard