Abstract
The TRANSWAY® software application is an adaptive, ontology-based toolset with collaborative agents, designed to assist the Deployment and Distribution Operations Center (DDOC) staff of the United States Transportation Command (USTRANSCOM) with the performance of movement planning tasks. The principal focus of the TRANSWAY® toolset is re-planning. The objective of TRANSWAY® is to provide DDOC operators with an intelligent planning engine that receives data from GTN and JOPES, and utilizes the existing Integrated Computerized Deployment System (ICODES) and Joint Forces Collaborative Toolkit (JFCT®; - formerly known as SEAWAY) agent-based applications as functional extensions.

With its initial and immediate focus on the DDOC and the JDDOCs in the theater, the TRANSWAY® toolset is designed as a component of a larger adaptive decision-support environment that is expected to evolve over the next several years. The broader focus of this umbrella decision-support environment is to support the Combatant Commands and USTRANSCOM in logistic planning and execution at the strategic and operational levels. In particular, this decision-support environment must provide the necessary connections between changing requirements and changes in the strategic flow of supplies and equipment to the Joint Operating Area. Joint operations are currently conducted in a framework that lacks key linkages
between the joint force *sustainment requirement* and the *supply and transport processes* that must satisfy this requirement.

Both the TRANSWAY® toolset and the more gradually evolving umbrella decision-support environment, of which it forms a component part, exemplify the planned transition of the existing USTRANSCOM Corporate Data Environment (CDE) to an intelligent Corporate Information-Centric Environment (CICE) with knowledge management capabilities. Compliant with the principles defined by the Department of Defense Net-Centric Data Strategy (Stenbit 2003), the proposed CICE architecture is designed to support the incremental implementation of progressively more intelligent and powerful tools operating within an adaptive collaborative decision-support environment that alerts its users to unforeseen events, assesses risks, identifies opportunities, generates alternatives, maintains in-transit visibility, facilitates time-critical re-planning, and collects lessons learned.

TRANSWAY® has been designed and implemented according to the principles of a Service-Oriented Architecture (SOA) and incorporates an integrated set of functional capabilities in the form of intelligent web-based thin-client and distributed thick-client tools. These tools, referred to as Knowledge Management Enterprise Services (KMES®), are *self-contained* software modules with powerful functional capabilities and clearly defined interface specifications. They are designed to be platform independent and may be configured to any particular execution environment. However, most importantly they are *reusable* as components of any system that has a need for their capabilities. Some of these services may have quite narrow capabilities such as the mapping of data imported from an external source to an internal information model or ontology, while others will incorporate larger functional domains such as the optimum routing of goods from multiple origins along alternative routes to multiple destinations.

The KMES® approach to software systems incorporates intelligent agent technology to analyze and categorize incoming signals and data, and then issue warnings and alerts as appropriate. The agents manipulate the incoming data within an internal information-centric representation framework (i.e., ontology) to publish statements of implication, and if so empowered, proceed to develop plans for appropriate action. Legacy data-centric systems can become clients of such an agent-based KMES® software environment through the use of *interoperability bridges* that map the data model in one system to the information model of the other and allow a two-way data exchange.

The notion of *service-oriented* is represented as much in the elements of the three distinct tiers (i.e., information, logic, and presentation) of the TRANSWAY® architecture, as it is in the functional capabilities of each KMES® component. Therefore, even the internal elements of a KMES® component communicate through standard interfaces as they provide services to each other. They are, in essence, decoupled software modules that can be replaced with improved modules as the technology advances. Each of these modules functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework.

In summary, KMES® modules are intelligent, self-contained software components that are capable of performing narrowly defined tasks within a net-centric system environment. They are distinguished from legacy software systems in five ways. First, they adhere to clearly defined application programming interface (API) specifications, but are otherwise decoupled from the service requesters that they interface with. Second, they are reusable and can be interfaced with systems that require the kinds of services that they provide. Third, they are able to provide their
services to both human users and other software systems. Fourth, their capabilities are enabled when they are configured (i.e., initialized) with the terminology and notions of the target domain. Fifth, they are self-contained and can be replaced by improved modules as the technology advances.

Like all KMES® components that CDM Technologies has developed over the past 12 years, the TRANSWAY® suite of planning and decision-support tools have been designed and implemented within the Integrated Cooperative Decision Making (ICDM) software environment. ICDM is an application development framework for distributed decision-support systems incorporating software agents that collaborate with each other and human users to monitor changes (i.e., events) in the state of problem situations, generate and evaluate alternative plans, and alert human users to immediate and developing resource shortages, failures, threats, and similar adverse conditions. A core component of any ICDM-based application is a virtual representation of the real world problem (i.e., decision-making) domain. This virtual representation takes the form of an internal information model, commonly referred to as an ontology. By providing context (i.e., data plus relationships) the ontology is able to support the automated reasoning capabilities of rule-based software agents.

Principal objectives that are realized to varying degrees by the ICDM Development Toolkit include: support of an ontology-based, distributed, information-centric system environment that limits internal communications to changes in information; ability to automatically push changes in information to clients, based on individual subscription profiles that are changeable during execution; ability of clients to assign priorities to their subscription profiles; ability of clients to generate information queries in addition to their standing subscription-based requests; automatic management of object relationships (i.e., associations) during the creation, deletion and editing of objects; support for the management of internal communication transmissions through load balancing, self-diagnosis, self-association and self-healing capabilities; and, the ability to interface with external data sources through interoperability bridges and ontological facades.

An Iterative Spiral Development Strategy has been adopted for the TRANSWAY® project. This software engineering approach fosters user participation throughout the design process, by bringing pre-release versions of the evolving application to potential users at relatively short time intervals throughout the development period. In this way users are encouraged to provide feedback on existing capabilities and recommendations for additional capabilities. The new demands on joint and coalition operations, which have precipitated a re-examination of deployment and distribution objectives and performance expectations, are likely to change significant aspects of joint theater distribution. Under these circumstances only an iterative development strategy will allow TRANSWAY® development to stay aligned with the corresponding changes in functionality requirements.

---

1 ICDM is a software development toolkit that is proprietary to CDM Technologies, Inc. and available to third parties under licensing agreements (Pohl et al. 2004).
# Table of Contents

1. Introduction and Requirements ......................................................... 9  
   1.1 Functional Requirements .......................................................... 9  
      1.1.1 Overall Functional Requirements ........................................ 9  
      1.1.2 Specific Application Requirements ..................................... 10  
   1.2 Technical Solution Approach ................................................... 11  
      1.2.1 The Military Logistics Planning Problem ............................... 12  

2. The Operational Capabilities of TRANSWAY® .................................. 15  
   2.1 Start-Up and Shut-Down ......................................................... 15  
   2.2 Editing Theater Elements ....................................................... 16  
   2.3 Displaying Reports ............................................................... 16  
   2.4 Creating Plans ........................................................................ 16  
   2.5 Typical TRANSWAY® Operational Scenario ............................... 17  

3. TRANSWAY® as a Suite of Intelligent Services ................................. 33  
   3.1 A Service-Oriented Architecture ............................................... 33  
      3.1.1 The Web Services Environment ........................................ 34  
      3.1.2 Adding meaning to Web Services ...................................... 34  
   3.2 Knowledge Management Enterprise Services .............................. 35  
      3.2.1 KMES® in a Net-Centric Architecture ................................. 36  
      3.2.2 A Typical KMES® Ontology ............................................. 39  
      3.2.3 The Capabilities of KMES® Agents ................................... 40  
      3.2.4 Benefits of the KMES® Approach ..................................... 40  
   3.3 ICDM: Philosophy and Principles .............................................. 41  
      3.3.1 Collaboration-Intensive ................................................... 42  
      3.3.2 Context-Oriented Representation ...................................... 44  
      3.3.3 Extensible and Adaptable ................................................ 45  
      3.3.4 Multi-Tiered and Multi-Layered ....................................... 46  
   3.4 The TRANSWAY® System Architecture ..................................... 47  

4. The Internal Information Model ..................................................... 51  
   4.1 The Main Domain ..................................................................... 52  
   4.2 The Time Domain .................................................................... 52  
   4.3 The Types Domain ................................................................... 53  
   4.4 The Constraints Domain .......................................................... 55  
   4.5 The Items Domain .................................................................... 55
The Environment Domain ................................................................. 57
The Organizations Domain ............................................................. 57
The Requirements Domain ............................................................. 58
The Alerts Domain ............................................................... 59
The Planning Domain .............................................................. 59

5. Utilized Information Services and Components ............................................. 61
   Facades ......................................................................................... 61
   Subscription Service ...................................................................... 62
   Persistence/Query Service ............................................................. 62
   Scenario Server ........................................................................... 63
   Imagery Service ........................................................................... 64

6. The TRANSWAY® Agents ................................................................. 67
   The Tabu Search Algorithm Approach .................................................. 67
      Tabu Search Memory Structures .................................................. 68
      Reactive Tabu Search ................................................................... 68
      Tabu Search for Constrained Problems ....................................... 69
      Open TS Framework .................................................................... 70
   The Agent Planning Approach in TRANSWAY® ........................................ 71
      Separation of Trip and Plan Generation ........................................ 71
      Selection of Search Methodology ................................................. 72
   Overall Architecture ........................................................................ 73
   The Planning Agent .......................................................................... 75
      System Architecture ...................................................................... 76
      Plan Creation Process .................................................................... 77
      Heuristic and Exhaustive Search Strategies ................................... 78
      Dynamic Planning ......................................................................... 78
   The Feasibility Agent ......................................................................... 79
   Truth Maintenance Agent .................................................................... 80
   Impediment Agent ............................................................................ 81
   Related Work in the Military Domain .................................................. 82
      Variations of the Vehicle Routing Problem ................................... 82
      Simulation versus Optimization Techniques .................................. 83
      Transshipment Problems ............................................................... 84

7. The Thick Client User-Interface ............................................................. 87
   Overall Architecture .......................................................................... 87
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>Generic Space Generator (GSG) Framework</td>
<td>88</td>
</tr>
<tr>
<td>7.3</td>
<td>GSG Mapping</td>
<td>90</td>
</tr>
<tr>
<td>7.4</td>
<td>Map Domain</td>
<td>91</td>
</tr>
<tr>
<td>7.5</td>
<td>Transfer Model Domain</td>
<td>92</td>
</tr>
<tr>
<td>7.6</td>
<td>Charts Domain</td>
<td>93</td>
</tr>
<tr>
<td>7.7</td>
<td>Agents Domain</td>
<td>94</td>
</tr>
<tr>
<td>8.</td>
<td>The TRANSWAY® Web Client</td>
<td>95</td>
</tr>
<tr>
<td>9.</td>
<td>Server Deployment</td>
<td>97</td>
</tr>
<tr>
<td>10.</td>
<td>Security</td>
<td>99</td>
</tr>
<tr>
<td>10.1</td>
<td>Network Security</td>
<td>99</td>
</tr>
<tr>
<td>10.2</td>
<td>Java Client Security</td>
<td>99</td>
</tr>
<tr>
<td>10.3</td>
<td>Web Client Security</td>
<td>100</td>
</tr>
<tr>
<td>10.4</td>
<td>Web Service Security</td>
<td>100</td>
</tr>
<tr>
<td>10.5</td>
<td>Server Operating System Security</td>
<td>100</td>
</tr>
<tr>
<td>10.6</td>
<td>Oracle Database Security</td>
<td>100</td>
</tr>
<tr>
<td>10.7</td>
<td>ActiveMQ Security</td>
<td>101</td>
</tr>
<tr>
<td>10.8</td>
<td>ApacheDS Security</td>
<td>101</td>
</tr>
<tr>
<td>10.9</td>
<td>Tomcat Security</td>
<td>101</td>
</tr>
<tr>
<td>11.</td>
<td>References and Bibliography</td>
<td>103</td>
</tr>
<tr>
<td>12.</td>
<td>Glossary</td>
<td>109</td>
</tr>
<tr>
<td>Appendix A: Design and Implementation of the Planning Agent</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>A-1</td>
<td>Problem Definition</td>
<td>113</td>
</tr>
<tr>
<td>A-2</td>
<td>Tabu Search Framework</td>
<td>114</td>
</tr>
<tr>
<td>A-3</td>
<td>Design of the Planning Agent</td>
<td>116</td>
</tr>
<tr>
<td>A-3.1</td>
<td>Create Moves</td>
<td>116</td>
</tr>
<tr>
<td>A-3.2</td>
<td>Select Move</td>
<td>117</td>
</tr>
<tr>
<td>A-3.3</td>
<td>Apply Move</td>
<td>118</td>
</tr>
<tr>
<td>A-3.4</td>
<td>Update Solution</td>
<td>119</td>
</tr>
<tr>
<td>A-4</td>
<td>Implementation of the Planning Agent</td>
<td>119</td>
</tr>
<tr>
<td>A-4.1</td>
<td>Tabu Search Framework (OpenTS)</td>
<td>119</td>
</tr>
<tr>
<td>A-4.2</td>
<td>Create Moves</td>
<td>120</td>
</tr>
<tr>
<td>A-4.2.1</td>
<td>Function <code>getTripForRequest()</code></td>
<td>121</td>
</tr>
</tbody>
</table>
A-4.2.2 Function createRequestTrip() and getTrip() .................. 122
A-4.2.3 Function createMoves() ............................ 124
A-4.3 Select Move ................................................. 125
A-4.4 Apply Move ................................................ 127
A-4.5 Update Solution .......................................... 127
A-5 Performance Analysis ......................................... 128
A-6 Discussion ....................................................... 129

Appendix B: The TRANSWAY® Data Dictionary .......................................................... 131
1. Introduction

The deployment and distribution responsibilities of USTRANSCOM call for intelligent collaborative tools in support of strategic and operational planning functions involving the sustainment and movement of forces. The sustainment requirement is generated at the operational level and is dynamic. It is composed of shifting priorities responding to changes in commander’s intent and changes in the operational situation. However, while commander’s intent and future plans normally drive the sustainment requirement, it is also possible for the reverse to occur. Unit movement and sustainment flow planning and execution monitoring is largely planned and executed at the strategic level, responding to ship and aircraft availability and other gross transportation factors only indirectly related to the changing operational priorities in the theater. Strategic flow planning and execution processes are focused on logistic efficiency and tonnage, while satisfying operational requirements is focused on logistic effectiveness (i.e., providing the right thing in the right quantity at the right place at the right time to the right units).

1.1 Functional Requirements

The functional vision of TRANSWAY® is to provide a set of integrated intelligent tools that will assist DDOC and JDDOC operators in the planning and re-planning of cargo shipments in support of unit movement and sustainment operations. In general terms this requires TRANSWAY® to facilitate the generation and modification of movement plans that involve multiple classes of heterogeneously packaged cargo and personnel, from several origins, moving on several kinds of transportation assets along various transportation routes, to multiple destinations. In addition, it is necessary for TRANSWAY® to provide access to the current state of the shipment as a contributor to in-transit visibility.

However, the TRANSWAY® suite of intelligent tools represent only a component of a larger adaptive decision-support environment that is aimed at bridging the current gap between joint force sustainment requirements and the supply and transport processes that must satisfy these requirements. All of these components are required to be interoperable at the information level, so that the exchange of data among the components will occur within an umbrella of context that is capable of supporting collaborative agents with automated reasoning capabilities.

1.1.1 Overall Functional Requirements

The functional capabilities of the umbrella decision-support environment in which TRANSWAY® exercises its focused planning and re-planning capabilities can be defined within the following general requirements:

1. Within a Joint Operating Area (JOA) the Joint Forces Commander (JFC) bears the responsibility for generating effective Courses of Action (COA) that: support the mission; can be executed at acceptable risk; are logistically supportable; and, can be delivered with the lift assets available.

General Requirement: The JFC requires intelligent decision support tools that directly translate force and mission plans into statements of logistic requirements with associated inventory-based opportunity costs and risks.
2. The resulting sustainment requirements are generated at the operational level, are dynamic, and are composed of shifting priorities responding to changes in JFC intent and changes in the operational situation. While JFC intent and future plans normally drive the sustainment requirement, it is also possible for the reverse to occur. Here, lacking tools to monitor sustainment flows, a sudden unplanned sustainment shortfall can result in the JFC rejecting a proposed COA after deciding that the inventory-based opportunity costs and associated risks are too great.

**General Requirement:** The JFC requires intelligent decision-support tools to assist in generating and re-generating the sustainment requirements based on intent and a proposed COA. In addition, these tools must support the rapid generation and assessment of an alternative COA when unexpected changes occur.

3. While the sustainment requirement is generated at the operational level, sustainment flow planning and execution monitoring is largely planned and executed at the strategic level, responding to ship and aircraft availability and other gross transportation factors only indirectly related to the changing operational priorities in the theater. Strategic flow planning and execution processes are focused on logistic efficiency and tonnage, while satisfying JFC operational requirements is focused on logistic effectiveness (i.e., providing the right thing in the right quantity at the right place at the right time to the right units).

**General Requirement:** Strategic planners require intelligent decision-support tools that detect changing sustainment priorities and automatically generate options that integrate transport assets, inventory availability, on-going operations, and so on.

4. Both strategic sustainment flows as well as distribution operations within the JOA are affected by the capacities and capabilities of the reception and onward staging facilities and infrastructure in the theater. The JFC’s plans for further movement and eventual distribution of supplies and equipment received at the Aerial Port of Debarkation (APOD) and Surface Port of Debarkation (SPOD) is dependent on incorporating a set of changing intra-theater lift, runway and/or pier capabilities, storage capacities, road conditions, etc., into the in-theater distribution planning. Strategic planners crafting airlift and sealift flows into theater APODs and SPODs have an identical need.

**General Requirement:** Both the strategic and operational planners require intelligent decision-support tools that are capable of integrating theater infrastructure capacities and characteristics (as well as changes to these) into sustainment and distribution plans, and projections.

### 1.1.2 Specific Application Requirements

The objective of TRANSWAY® is to specifically address the re-planning functions that must be performed in response to a dynamically changing distribution environment in the joint theater AOR. In this respect TRANSWAY® must:

1. Integrate the logistic-related characteristics of the theater infrastructure into JFC planning tools so that distribution and delivery plans continuously reflect changing port, airfield, rail, road, and marshalling capabilities.
2. Display current in-transit and scheduled status of supplies and equipment vis-à-vis the required delivery schedule, which reflects JFC operational planning, and automatically identify the status of priority items.

3. Provide capabilities for the automatic generation of alerts and warnings indicating those items, which will not meet currently required delivery dates contained in the JFC delivery schedule.

4. Provide for the automatic generation of options in the event delayed delivery is indicated, including the use of alternate transportation and substitute items if appropriate.

These conceptual requirements can be defined in more detail under the functional categories of mediation (i.e., connectivity to external systems), collaboration, adaptation, information presentation, and deployment considerations.

**Mediation capabilities:** TRANSWAY® must provide the capability to react to changing events in the theater by utilizing data from existing strategic and operational logistics systems. This includes provision for an open system architecture that facilitates the linkage to existing DoD deployment and sustainment systems serving as data sources and recipients of generated plans and assessments.

**Collaboration capabilities:** TRANSWAY® must provide collaborative agent assistance in the development or modification of movement plans within the theater AOR by: incorporating load-planning techniques as they apply to distribution operations; supporting load sequencing of multiple cargo types for discharge at multiple destinations; accommodating the full range of transportation planning and execution functions; providing a dynamic movement planning and re-planning toolset; supporting multi-modal load-planning; supporting onward movement of military equipment, supplies and personnel from APODs and SPODs to any node in the theater AOR; providing a seamless transition from strategic lift assets to theater lift assets; and, supporting movement planning by transportation mode.

**Adaptation capabilities:** TRANSWAY® must provide the operator with the ability to modify planning parameters in order to develop or modify plans that reflect the current conditions of the theater. Specifically, this includes parametric data and user-defined operational variables.

**Information presentation capabilities:** TRANSWAY® must be capable of presenting in-transit visibility information and up-to-date logistical situation awareness at all echelons through secure web-based Internet user access.

**Deployment access:** TRANSWAY® must provide a systems framework that will accommodate deployment at all echelons within the USTRANSCOM DDOC, at service specific installations, at sea ports and airports, and at theater JDDOCs.

### 1.2 Technical Solution Approach

These requirements dictate the design and configuration of TRANSWAY® as a set of intelligent collaborative tools supporting operators performing planning and re-planning tasks in a dynamically changing decision-making environment. The open, service-oriented architecture of
TRANSWAY® allows these capabilities to be progressively extended. Current functional capabilities include:

- Intelligent decision-support tools that detect changing sustainment priorities and automatically generate options that integrate transport assets, inventory availability, and ongoing operations.
- Intelligent decision-support tools capable of integrating theater infrastructure capacities and characteristics (as well as changes to these) into sustainment and distribution plans, and projections.
- Intelligent decision-support tools that ensure continuous visibility of both the dynamic sustainment requirements and the strategic sustainment plans generated in response to these requirements.

For a computer to interpret data it requires an information structure that provides at least some level of context. This can be accomplished utilizing an ontology of objects with characteristics and a rich set of relationships to create a virtual version of real world situations and provide the context within which agent logic can automatically operate. In TRANSWAY® the underlying ontology is divided into logical domains that can be described using the Unified Modeling Language (UML) methodology. Within each domain exist definitions of the various concepts and entities relevant to the representation and analysis of key aspects of each domain. Classes located within package symbols are defined within that domain. These classes may relate to classes defined in other domains through either inheritance or associations. In both cases, referenced classes are identified by their symbols existing outside the primary package symbol with some type of relationship symbol connecting them to package elements. Domains themselves may be related to each other in either a sibling or parent/child relationship. Such connections are an indication of the particular scope and inter-domain visibility.

TRANSWAY® includes two kinds of agents with strategic and operational planning and replanning capabilities, respectively. The strategic planning agents are based on the Tabu Search genetic algorithm and the operational planning agents are rule-based and implemented in Java. Tabu Search is a local search method for exploring a solution space. In the TRANSWAY® implementation of the Tabu genetic algorithm the solution space is every possible planning recommendation. Starting from an initial empty plan, new plans are generated and immediately evaluated based on a merit function. The highest rated plan then becomes the new incumbent best solution, followed by a repetition of the same procedure. Once some ending criterion has been reached the algorithm may stop and report the best solution that it has found or, as in the current version of TRANSWAY®, reporting may occur on a continuous basis as better and better solutions are found.

1.21 The Military Logistics Planning Problem

The military domain presents a challenge for traditional supply chain optimization techniques, because problems are dynamic and cannot be modeled hierarchically. Flow networks are a common technique for supply chain management, but flow networks cannot solve highly

---

This Section is based largely on the following thesis: Weber B. (2006); ‘Optimization of Strategic and Operational Planning Utilizing Tabu Search’; Master degree thesis, Department of Computer Science, California Polytechnic State University, San Luis Obispo, California, June.
constrained problems. Simulation has been utilized for supply chain optimization, because simulation is flexible and suitable for dynamic environments. However, simulation discovers less intelligent solutions than optimization techniques. Optimization approaches have been independently applied to strategic and operational levels of planning. Solving these problems separately may result in inefficient solutions, because the problems are highly coupled.

Optimization techniques have not been applied towards simultaneously solving strategic and operational levels of planning. The size of the problem requires sacrificing optimality for a practical run time. The TRANSWAY® application utilizes a Tabu Search algorithm that considers the interconnections between assets at the strategic and operational levels of planning. The algorithm utilizes an approximated objective function, enabling the algorithm to separate the optimization process from the constraint verification process.

Traditional supply chain techniques have been shown to be inefficient when applied in the military domain. Serious shortcomings have illustrated the need for improved logistical processes in military operations such as Desert Storm (Kaminski 1995) and Allied Force (Brooks 2000). More recently, the shortfall of add-on-armor in Iraq (Bowman 2003) confirmed supply chain problems. It is necessary to discard the just-in-case inventory approach (Schrady 1999) and move to a rapid and reliable transportation process that provides time definite delivery to customers (Crino et al. 2002). Current planning techniques include manual planning and simulation. However, Semet and Taillard (1993) have demonstrated that solutions generated by application of the Tabu Search algorithm are significantly superior to those obtained by hand.

Supply distribution in the military sector differs significantly from supply distribution in the commercial sector. Problems in the commercial domain are often represented hierarchically and analyzed using supply chain techniques to trace the throughput at each node (Beamon 1998). However, representing problems hierarchically in the military domain may result in the inefficient utilization of resources. Problems in the commercial domain are less dynamic than problems in the military domain, because military problems operate in hostile environments. Therefore, military planners apply countermeasures against possible threats. Also, security of transportation assets is not as vital in the commercial domain.

Military supply distribution is divided into strategic, operational and tactical levels of planning. Operational planning consists of the allocation of supplies and personnel within a theater. A theater is defined as a geographical area of operation outside of the continental United States (Crino et al. 2002). Strategic planning consists of the distribution of supplies, personnel and transportation assets between different theaters of operation. The tactical level of planning specifies the movement of supplies from locations within a theater to individual units. TRANSWAY® currently considers only the strategic and operational levels of planning.

Existing models for military distribution utilize mainly simulation techniques (Wu et al. 2003) even though simulation has several limitations. Rule-based models satisfy constraints, but do not optimize the utilization of resources. In the military domain, it is necessary to generate alternate plans to deal with contingencies. However, simulations are deterministic and always generate the same solution. Applying a simulation to a new problem may require changes to the rule set (Wu et al. 2003). The main advantages of simulation are that resulting solutions are generated quickly and guaranteed not to violate constraints. However, simulation techniques produce low quality solutions for the military domain.
TRANSWAY® utilizes a Tabu Search algorithm for simultaneously solving strategic and operational levels of planning, with the contention that the consideration of both levels of planning allows a more efficient utilization of resources due to the interdependencies between the problems. The empirical verification of the hypothesis that for strategic and operational planning optimization outperforms current simulation techniques was validated by the thesis research mentioned above (see footnote 2).
2. The Operational Capabilities of TRANSWAY®

An operational view of the planning, re-planning and execution simulation capabilities of TRANSWAY® is depicted in Figure 2.1 in terms of four principal activity areas: starting up and shutting down the system; defining and editing the environmental context of the application domain; displaying reports; and, creating plans based on initial and changing conditions.

2.1 Start-Up and Shut-Down

After start-up, TRANSWAY® provides options for importing sets of objectified theater elements (e.g., Supply Support Activity (SSA) nodes, Ports of Embarkation and Debarkation (POE and POD), route legs, and so on) presenting their associated graphics as layers on automatically geo-referenced maps (Vector Product Format (VPF), Compressed ARC Digitized Raster Graphics (CADRG), and Controlled Image Base (CIB)). These objects, which may be alternatively entered and edited by the user, are represented as instances of the internal ontology providing context for the reasoning capabilities of the TRANSWAY® decision-support agents.

A second component of the initialization process is the start-up of the TRANSWAY® Agent Tier. This activity initializes the agents with appropriate context parameters. Once initialized the agents register their individual information interests with the Subscription Service. Upon satisfaction of such interests, subscribers (i.e., agents, GUI components, etc.) are automatically notified of data changes and will react according to their business rules.
2.2 Editing Theater Elements
This activity adds or changes characteristics of the theater. Editable theater elements include the following:

- **Requirements:** This activity adds a new list of requirements or changes an existing list by adding a new requirement, modifying an existing requirement (i.e., change quantity, change delivery time, change destination, etc.), or deleting an existing requirement.

- **Supplies:** This activity adds supplies to the environment or deletes existing supplies.

- **Conveyances:** This activity adds, modifies, or removes conveyances from the environment.

- **Routes:** This activity adds new routes to the environment or modifies existing routes in the environment.

- **Threats or impediments:** This activity adds a threat or impediment indicator to the environment by adding a polygon to represent the area that is affected by the threat or impediment. While impediments or effects are typically created and modified through agent-based analysis, they can also be manually entered and edited by the operator.

2.3 Displaying Reports
Reports provide users with information about various aspects of the evolving operations. Among the set of reports available in TRANSWAY® are inventory, cargo, conveyance, and agent reports. Inventory reports show users the availability and status of supplies. Cargo reports show the status of outgoing supplies. Conveyance reports show the availability and status of conveyances at nodes. Produced through agent-based analysis, agent reports communicate various outstanding issues and recommendations to operators and other system components (e.g., other agents, external systems, etc.). Some of these reports identify issues with existing or evolving delivery plans, including inventory deficiencies or shortfalls and threats or impediments that impact the ability to execute an existing plan.

2.4 Creating Plans
This activity involves the specification or selection of a set of requirements that need to be satisfied along with various criteria that can help shape the resulting plan (e.g., priority for the conservation of transports or supply missions, strengthening or relaxation of certain delivery windows, and transport and supply center selection preferences). Once such criteria have been adequately specified, the agent-based planning process can be initiated. The effectiveness of resulting plans can be compared through various reports.

Apart from specific planning criteria, plans are created based on the current state of available supplies, requests for supplies (i.e., requirements) that need to be satisfied, the priority of these requirements, the availability of conveyances, the nature of alternative routes, and the existence of any threats or impediments that may influence the selection of routes. Throughout the planning and plan-monitoring process agents identify and communicate various issues (i.e., inventory and transport shortfalls, etc.) through the creation of agent alerts. An agent alert is also created to indicate the completion of the planning activity.
2.5 Typical TRANSWAY® Operational Scenario

The main TRANSWAY® screen (Figure 2.2) is divided into two principal areas. On the left side, moving from the top down, below the main option bar the user will find: three agent icons; objects that may be placed on top of the map (the right side of the screen); a tree-structure that provides quick and convenient access to the data that the system is currently populated with; and, at the bottom a command window for the Tabu agent. On the right side of the screen is a geo-referenced map that allows the user to pan to any part of the world and, subject to the availability of maps, zoom down to street level if desired. Objects representing nodes (e.g., SAAs, APODs, etc.), route segments, impediments, and areas of interest may be moved from the left side of the screen to the right side by simple *click to locate* actions. Alternatively, the user may specify latitude-longitude locations and the selected object will be automatically placed on the map in the correct location. These objects, whether entered by the user or pre-initialized in the system, have attributes that relate to the internal representation (i.e., ontology) of TRANSWAY® and provide the necessary context for automated agent actions.

![Figure 2.2: Main TRANSWAY® screen](image)

TRANSWAY® is by no means limited to the current set of attributes. With the contractual goal of this first version of a prototype system to demonstrate the typical capabilities of an ontology-based multi-agent system, attributes were selected in a fairly generic fashion based on the feedback that the development team received during early demonstrations, perusal of military documents, and in-house experience with other logistic planning systems such as the Integrated Computerized Deployment System (ICODES) and the Joint Forces Collaborative Toolkit (JFCT®).
Figure 2.3: Summary of supplies and available conveyances at supply centers

The report shown in Figure 2.3 provides a summary of supplies (short tons) and available conveyances (i.e., fixed wing aircraft, helicopters, ships, and trucks (in convoys)) at most supply centers currently initialized in the system for this particular demonstration scenario. Details of
supplies at Charleston and Al Udeid are shown in Figures 2.4 and 2.5 (in terms of supply Class, number of pallets, number of items per pallet, and short tons), respectively.

Figure 2.4: Details of supplies at Charleston

Figure 2.5: Details of supplies at Al Udeid
Figure 2.6 provides information about the air channels and sea routes that the system has been initialized with for this particular demonstration scenario. In each case the two end-points and the distance in nautical miles is indicated.
Detailed information about the current compliment of conveyances can be obtained by selecting the appropriate report. Typical examples for various fixed wing aircraft, trucks and ships are shown in Figures 2.7 to 2.12, below. The reason that the speed and bearing attributes in each table are zero is because the conveyances are not currently in-transit.

Figure 2.7: Boeing 747 aircraft attributes

Figure 2.8: C5 aircraft attributes
Figure 2.9: C17 aircraft attributes

Figure 2.10: C130 aircraft attributes
<table>
<thead>
<tr>
<th>Conveyance Type</th>
<th>Speed</th>
<th>Bearing</th>
<th>Conveyance Type</th>
<th>Pallet Positions</th>
<th>Minimum Coasting Speed</th>
<th>Maximum Range</th>
<th>Location</th>
<th>Unavailability</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
<tr>
<td>20-Ft Flatbed Truck</td>
<td>0 mph</td>
<td>0</td>
<td>20-Ft Flatbed Truck</td>
<td>4,000 kg</td>
<td>40.0 m/s</td>
<td>400.0 km</td>
<td>2,000 km</td>
<td>E</td>
</tr>
</tbody>
</table>

**Figure 2.11: Truck convoy attributes**

<table>
<thead>
<tr>
<th>Conveyance Type</th>
<th>Speed</th>
<th>Bearing</th>
<th>Conveyance Type</th>
<th>Pallet Positions</th>
<th>Minimum Coasting Speed</th>
<th>Maximum Range</th>
<th>Location</th>
<th>Unavailability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
<tr>
<td>Vessel</td>
<td>0 mph</td>
<td>0</td>
<td>Vessel</td>
<td>1000 kg</td>
<td>17.3 m/s</td>
<td>20,000 kg</td>
<td>20,000 kg</td>
<td>E</td>
</tr>
</tbody>
</table>

**Figure 2.12: Typical ship attributes**
A typical request for add-on-armor is shown in Figure 2.13. It requires deliver to Al Udeid, with a high priority and an earliest and latest time for delivery window of 25 to 31 December 2005.
To fulfill the request for the shipment of *add-on-armor* to Al Udeid (Figure 2.13) the user activates the Tabu agent and selects the appropriate *requirement* from the displayed Requirement Lists (Figure 2.15). In this case the Al Udeid *requirement* is Requirement List 1. Since the Tabu
agent has the ability to continue its search for an optimum delivery plan even after it has found a way of satisfying the requirement, the user has the option of either setting a maximum time for the planning activity (Figure 2.16) or allowing the agent to continue until all alternatives have been explored. Of course it is not expected that the user would ever want to wait for that length of time and therefore the option for the user to simply stop the agent is available. In future versions of TRANSWAY®, particularly if the Tabu agent were to be implemented in an opportunistic mode (i.e., in a manner that would activate the planning process without user involvement as soon as the conditions on which an existing plan were originally based have changed), it would be a relatively simple matter to restrict the extensiveness of the search for an optimum plan. For example, the search could be automatically aborted if after either a specified period of time or a given number of generated plans no better plan has been found.

Figure 2.18: Weather impediment

Figure 2.19: Impediment agent alert
For the completed plan the route is shown in Figure 2.17 by means of a red line. Next the user enters an impediment in the form of an adverse weather report that essentially eliminates Glasgow as a refueling stop (Figure 2.18). Immediately, the Impediment agent alerts the user and suggests that re-planning is in order (Figure 2.19). Again, also in the case of impediments, this first version of TRANSWAY® provides only one type of generic impediment (i.e., a weather condition), with the objective of demonstrating the kinds of causes that would require re-planning that could be easily implemented in subsequent versions of the system, based on user preferences and priorities.

![Image of a computer interface showing delivery plans](image)

**Figure 2.20:** Summary of deliveries for the first and second plans

To initiate a re-planning action the user proceeds in the same manner as described previously for the generation of the first plan (Figures 2.15 to 2.17). The user will notice that during the generation of each plan the routes that are being explored by the Tabu agent are dynamically indicated on the map display. Temporarily displayed green lines indicate drop-off points that are being considered. Red lines indicate actual delivery routes with the thickness of the red line
providing a proportional indication of the volume of supplies being transported along that particular route. Summary lists of the deliveries involved in both plans are shown in Figure 2.20. Even though this first test-bed version of TRANSWAY® is purposely limited in scope it does allow the user to explore the details of each delivery plan (i.e., start and end locations, conveyances and routes used, start and end times, and duration of each trip), as shown in Figures 2.21 to 2.24.
Figure 2.23: Typical drill-down details of the second plan

Figure 2.24: Typical drill-down details of the second plan

29
Figure 2.25: Comparison of conveyances needed in support of the first and second plans

Figure 2.26: Comparison of overall lift requirements for the first and second plans
Apart from the ability of the user to drill down into the details of each delivery plan there are a number of comparative graphical reports available, such as the utilization of specific conveyances by each plan shown in Figure 2.25 and the number of conveyances that are required to support each plan over time shown in Figure 2.26.

**Figure 2.27:** Departures from Charleston by conveyance type

**Figure 2.28:** Departures from Dover

**Figure 2.29:** Departures from Al Udeid
Figures 2.27 to 2.29 show examples of conveyance departures from the Charleston, Dover and Al Udeid APODs, respectively. Similar reports are available for cargo transfers by date (Figures 2.30 to 2.31) in terms of what was lifted yesterday, the current inventory, and what is planned to be lifted during the next 72 hours. In this way the user is able to determine the expected volume of shipments from any particular APOD on a daily basis. The dates selected for the example bar chart reports shown in Figures 2.30 and 2.31 are December 23 to 26, 2005.

Figure 2.30: Typical cargo transfer history, status, and 72-hour projections

Figure 2.31: Typical cargo transfer history, status, and 72-hour projections

Again, these reports are intended to be examples of the kind of information that can be made available by TRANSWAY®. The development team will be guided by feedback from users in future development cycles. The reporting capabilities of the system can be easily extended in any direction within the constraints of data availability.
3. TRANSWAY® as a Suite of Intelligent Services

This chapter and all subsequent chapters describe TRANSWAY® from a technical perspective. The TRANSWAY® system is dissected into its principal Knowledge Management Enterprise Services (KMES®) components and examined in terms of design methodology and exploited technologies. In addition, fundamental architectural patterns such as its Service-Oriented Architecture (SOA) and internal context representation model (i.e., ontology domains) are described along with key component interaction models.

However, before embarking on a technical journey into the inner-workings of the TRANSWAY® agent-based decision-support system, a discussion of the underlying software design philosophy on which TRANSWAY® is founded is in order. Collectively, these qualities are derived from the KMES® approach and the ICDM-based principles employed by both CDM Technologies and the Collaborative Agent Design Research Center (CADRC) in the design and development of ontology-based multi-agent systems.

3.1 A Service-Oriented Architecture

The notion of service-oriented is ubiquitous. Everywhere we see countless examples of tasks being performed by a combination of services, which are able to interoperate in a manner that results in the achievement of a desired objective. Typically, each of these services is not only reusable but also sufficiently decoupled from the final objective to be useful for the performance of several somewhat similar tasks that may lead to quite different results. For example, a common knife can be used in the kitchen for preparing vegetables, or for peeling an orange, or for physical combat, or as a makeshift screwdriver. In each case the service provided by the knife is only one of the services that are required to complete the task. Clearly, the ability to design and implement a complex process through the application of many specialized services in a particular sequence has been responsible for most of mankind’s achievements in the physical world. The key to the success of this approach is the interface, which allows each service to be utilized in a manner that ensures that the end-product of one service becomes the starting point of another service.

In the software domain these same concepts have gradually led to the adoption of Service-Oriented Architecture (SOA) principles. While SOA is by no means a new concept in the software industry it was not until web services came along that these concepts could be readily implemented (Erl 2005). In the broadest sense SOA is a software framework for computational resources to provide services to customers, such as other services or users. The Organization for the Advancement of Structured Information (OASIS) defines SOA as a “... paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” and “...provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects with measurable preconditions and expectations”. This definition underscores the fundamental intent that is embodied in the SOA paradigm, namely

---

3 CDM Technologies, Inc. is the commercial arm of the Collaborative Agent Design Research Center (CADRC) at California Polytechnic State University (Cal Poly), San Luis Obispo.

4 OASIS is an international organization that produces standards. It was formed in 1993 under the name of SGML Open and changed its name to OASIS in 1998 in response to the changing focus from SGML (Standard Generalized Markup Language) to XML (Extensible Markup Language) related standards.
To be as flexible as possible a SOA environment is highly modular, platform independent, compliant with standards, and incorporates mechanisms for identifying, categorizing, provisioning, delivering, and monitoring services.

### 3.1.1 The Web Services Environment

Web services are a particular implementation of the SOA paradigm. According to the World Wide Web Consortium (W3C) a web service may be defined as “… a software application identified by a Uniform Resource Identifier (URI), whose interfaces and bindings are capable of being defined, described, and discovered by XML artifacts”\(^5\). Currently most web services interact with other services or users utilizing the Hyper-Text Transfer Protocol (HTTP) to exchange XML-based messages defined in the Service Oriented Architecture Protocol (SOAP). The SOAP standard defines an XML language and a set of rules for formatting objects and data that are independent of the programming language, operating system, and hardware platform.

Existing web service environments such as Microsoft’s ‘.Net’ software (Thai 2003, Chappell 2006) typically comprise a web server that utilizes HTTP for communication, UDDI as part of the standard definition of web service registries, a registry that already contains an entry for the accessing application, and any number of web services designed to facilitate some of the operations that the accessing application may wish to perform. In this respect, current web service environments rely on the notion of predefined registrations and discovery and do not support the notion of opportunistic discovery. UDDI, an international standard that defines a set of methods for accessing a registry, is structured to provide information about organizations, such as: who (about the particular organization); what (what services are available); and, where (where are these services available). However, UDDI does not provide descriptions of the available services in a semantic form that can be automatically interpreted by software (e.g., software agents), rather the descriptions are hard-coded or subject to human interpretation.

Communication between an application and a web server is almost always initiated by the application (i.e., the application sends a request and the web server sends a response). Specifically, the Uniform Resource Locator (URL) contains an identification of the particular web server to be used. This web server then finds the HTML page that corresponds to the URL and returns that page in the response. Immediately after the response has been sent the connection between the application and the web server is terminated and only reactivated if another response is requested. In this way a web server is able to handle many concurrent requests from applications.

### 3.12 Adding Meaning to Web Services

There are several reasons why computer software, and therefore web services, must increasingly incorporate more and more intelligent capabilities (Pohl 2005). Perhaps the most compelling of these reasons relates to the current data-processing bottleneck. Advancements in computer technology over the past several decades have made it possible to store vast amounts of data in electronic form. Based on past manual information handling practices and implicit acceptance of the principle that the interpretation of data into information and knowledge is the responsibility of the human operators of the computer-based data storage devices, emphasis was placed on storage efficiency rather than processing effectiveness. Typically, data file and database

---

\(^5\) See web site at: [http://www.w3.org/TR/wsa-reqs#IDAIO21B.](http://www.w3.org/TR/wsa-reqs#IDAIO21B).
management methodologies focused on the storage, retrieval and manipulation of data transactions, rather than the context within which the collected data would later become useful in planning, monitoring, assessment, and decision-making tasks.

Several years before the advent of the Internet and the widespread promulgation of SOA concepts the CADRC Center at Cal Poly started building distributed software systems of loosely coupled modules that were able to collaborate by subscription to a shared information model. Today, CDM Technologies’ KMES® components are based on the same foundational principles to enable them to function as decoupled services. These principles and corresponding capabilities include:

- An internal information model that provides a usable representation of the application domain in which the service is being offered. In other words, the context provided by the internal information model must be adequate for the software application (i.e., service) to perform as a useful adaptive set of tools in its area of expertise.
- The ability to reason about events within the context provided by the internal information model. These reasoning capabilities may extend beyond the ability to render application domain related services to the performance of self-monitoring maintenance and related operational efficiency tasks.
- Facilities that allow the service to subscribe to other internal services and understand the nature and capabilities of these resources based on its internal information model.6
- The ability of a service to understand the notion of intent (i.e., goals and objectives) and undertake self-activated tasks to satisfy its intent. Within the current state-of-the-art this capability is largely limited by the degree of context that is provided by the internal information model.

Additional capabilities that are not yet able to be realized in production systems due to technical limitations, but have been demonstrated in the laboratory environment, include: the ability of a service to learn through the acquisition and merging of information fragments obtained from external sources with its own internal information model (i.e., dynamically extensible information models); extension of the internal information model to include the internal operational domain of the software application itself and the role of the service within the external environment; and, the ability of a service to increase its capabilities by either generating new tools (e.g., creating new agents or cloning existing agents) or automatically searching for external assistance.

### 3.2 Knowledge Management Enterprise Services

Knowledge Management Enterprise Services (KMES®) are self-contained software components that offer their capabilities as services to external service requestors. Whereas in a SOA-based software system the available services normally operate at a lower system level as enablers of higher level functional capabilities, a KMES® component is the incarnation of one or more of those functional capabilities. In other words, KMES® components are services that operate at the functional level of the application domain.

---

6 This must be considered a minimum system capability. The full implementation of a web services environment should include facilities that allow a service to discover other external services and understand the nature and capabilities of these external services.
They are designed to be platform independent and adaptable to a variety of applications. It is this adaptability that promotes their high degree of reusability. Some KMES® components may have quite narrow capabilities such as the reformatting of weather data into a software interpretable weather forecast, while others will incorporate larger functional domains such as the optimum routing of goods from multiple origins along alternative routes to multiple destinations in TRANSWAY®. However, all of the services that operate within a given application domain are closely aligned to the knowledge context of that domain by sharing the same information model. While the software code of a KMES® component is reusable, its internal information model needs to be reconfigured as it is moved from one application domain to another. This ensures that the KMES® services within any particular application environment are able to exchange data within the functional context of that environment.

For example, in the transportation domain of TRANSWAY® the optimum routing of goods from multiple origins along alternative routes to multiple destinations would include the following KMES® components, providing their services within the common context of a shared information model:

- Conveyance load-planning (i.e., ships, barges, trucks, railcars, and aircraft of various types).
- Packaging of different kinds of shipping units (e.g., containers, pallets).
- Storage management in marshalling yards and warehouses.
- Route planning and re-planning.
- Map-based presentation for geospatial tracking.
- Scheduling.
- Interoperability bridges to external data feeds and other applications.
- Graphical and textual report generators.

This is in stark contrast to the large software systems that have been developed in the past and that invariably lead to a stove-piped architecture with almost insurmountable interoperability problems. Typically, in the case of these legacy systems the above functional capabilities have required the development of several systems with considerable duplication (e.g., user-interfaces, persistence facilities, and report generation) and largely incompatible data schemas.

3.21 KMES® in a Net-Centric Architecture

The expressive, context-rich representation upon which many KMES® capabilities are built together with the significant potential for higher levels of decision-support lends itself to incorporation of intelligent agent technology. When equipped with such enabling features, agents can collaborate with users to assist in formulating solution alternatives, compare and contract their associated costs, and aid in successful execution through constant monitoring and the performance of necessary mediation. For example, agents in the military logistical domain of TRANSWAY® can receive status reports, track shipments, incorporate suitable and available assets in plans, and provide appropriate updates on location and security risks. Others may track the path of incidence and provide appropriate graphic and textual updates for action. Finally, agents can interpret incoming signals, identify significant events (i.e., changes), and modify proposals to meet the changing situation as it develops. The vision of such intelligent agents is quite compelling and now generally believed to be a critical component for successfully harnessing the increasing complexities of a net-centric environment.
Existing data-centric systems lacking the adaptive, interoperability characteristics described above can be integrated into such an agent-empowered KMES® software environment through the use of interoperability bridge facilities that effectively map the data model in one system to the information model of the other. This allows for a meaningful bi-directional interaction and exchange of data in context. Such bridges have been successfully demonstrated by military organizations for linking legacy data-centric systems to intelligent command and control systems (Pohl et al. 2001). The technology is inherently scalable and allows for the efficient and effective interconnection of multiple participants within a heterogeneous net-centric environment.

Conceptually, an intelligent net-centric software environment typically requires the seamless integration of a KMES®-based information management facility with existing data sources. This can be achieved with an information-centric architecture that consists essentially of two components (Figure 3.1): a data-centric Data Capture and Integration Layer that incorporates linkages to existing data sources; and, an Intelligent Information Management Layer that overlays the data layer and utilizes software agents with automatic reasoning capabilities, serving as decision-support tools.

The Intelligent Information Management Layer architecture (Figure 3.1) utilizes intelligent software agents capable of collaborating with each other and human operators in planning, re-planning, monitoring, and associated decision-support environments. Typically such intelligent systems are based on software development frameworks, such as the ICDM (Integrated Cooperative Decision Making) and TIRAC® (Toolkit for Information Representation and Agent Collaboration) software development frameworks used by CDM Technologies and the CADRC
Data Capture and Integration Layer: The bottom layer of the system takes the form of an operational data store and/or Data Warehouse, implemented within a commercial off-the-shelf relational database management system (RDBMS). This repository integrates data extracted on a periodic basis from several external sources into a common data schema. Although not a requirement, the design of the data schema is typically closely modeled on the structure of the ontology of the Intelligent Information Management Layer to minimize the required data-to-information and information-to-data mappings between these two system layers. Further, to facilitate an object-oriented environment, content managed by the Data Capture and Integration Layer is exposed to its information-oriented clients (e.g., KMES®-based environments) as objects rather than relational tables. Translation between these two forms is typically accomplished through employment of some form of Object Relational Mapping (ORM) technology.

In conformity with normal enterprise data management practices the Data Capture and Integration Layer incorporates the following four characteristics:

• It is subject-oriented to the specific business processes and data domains relevant to the application area (e.g., goods movement across national borders or tactical command and control in a military theater).

• It is integrated so that it can relate data from multiple domains as it serves the data needs of the analysis functions performed by collaborative agents in the Intelligent Information Management Layer.

• It is periodically synchronized with events and changes occurring in the external data sources from which it derives its content.

• It is time-based to support the performance of analyses over time, for the discovery of patterns and trends.

A multi-tier architecture is used to logically separate the necessary components of the data layer into levels. The first tier is the RDBMS, which ensures the persistence of the data level and provides the necessary search, persistence, and transaction management capabilities. The second tier is the service level, which provides the interface to the objectified data level and at the same time supports the data access requests that pass through the mapping interface from the Intelligent Information Management Layer to the Data Capture and Integration Layer. It is designed to support request, response, subscribe, and publish functionality. The third tier is the control level, which routes information layer and user requests to the service level for the update, storage and retrieval of data. Finally, a view layer representing the fourth tier serves as a graphical user-interface for the Data Capture and Integration Layer.

Information Management Layer: The Intelligent Information Management Layer consists of KMES® components in the form of a group of loosely coupled and seamlessly integrated decision-support tools. The core element of each KMES® component is typically an ontology that provides a relationship-rich and expressive model of the particular domain over which the KMES® capability operates. Normally, KMES® components are based on a three-tiered architecture incorporating technologies, such as distributed-object servers and inference engines, to provide a framework for collaborative, agent-based decision-support that offers developmental efficiency and architectural extensibility. The multi-tiered architecture clearly distinguishes between information, logic, and presentation. Most commonly an information tier
consists of a collection of information management servers (i.e., information server, subscription server, etc.) providing domain-oriented access to objectified context, while a logic tier houses communities of intelligent agents, and a presentation tier is responsible for providing meaningful interfaces to human operators and external systems.

### 3.2.2 A Typical KMES® Ontology

This section discusses a portion of a domain-centric ontology upon which a particular KMES® capability may operate. In the example provided below, the underlying KMES® ontology is divided into several somewhat related domains (Figure 3.2). While some of these domains describe application-specific events and information (e.g., goods movement transactions, shipping routes, and so on) others describe more general, abstract notions (e.g., event, threat, view, privacy). The goal in developing such an ontology is to abstract general, cross-domain notions into high-level, extensible domain models. As such, these descriptions can be refined and specialized across several application sub-domains. In other words, more domain-specific, concrete notions can be described as extensions of these abstract models.

![Figure 3.2: Typical ontology domains within a military application area.](image)

Accordingly, a KMES® ontology includes several primary meta-characteristics. Through mechanisms such as inheritance as well as the application of underpinning analysis patterns, these meta-characteristics can be propagated to more specific ontological components. To illustrate, the simple application of inheritance allows, for example the abstract characteristic of something being trackable to be propagated into more specialized entities. Applied to this logistics example, if such a trackability notion is introduced at the physical.Mobile level then, through inheritance, any entity that is a kind of physical.Mobile automatically receives the property of being trackable. Taking this example further, a second meta-characteristic may relate to the dispensability of an item. If this property is represented at the physical.Item level then, similar to the trackable characteristic, anything that is a kind of a physical.Item automatically receives the quality of being dispersible or suppliable. In addition, as an extension of
physical Mobile, such suppliable items are also trackable. It can be readily seen that together these two meta-characteristics provide an effective foundation for propagating fundamental notions to hierarchically related definitions. Although inheritance can be a useful mechanism for the propagation of fundamental characteristics to more specific classifications, an even more powerful, and oftentimes less restrictive, technique is the application of extensible analysis patterns. Such patterns offer adaptable model fragments, thereby providing a fundamental definition of the notion being represented in the form of an extensible model architecture for applying this underpinning concept to other elements of a domain model. Such patterns typically employ a role metaphor, where elements of a model may essentially play the role of something embodying the fundamental notion or characteristic.

### 3.2.3 The Capabilities of KMES® Agents

KMES® components equipped with intelligent agents may employ a variety of framework technologies and reasoning paradigms to execute their agent-based logic. Regardless of the specific agent technology employed, their capabilities can exist at a monitoring, largely reactive level, or at a higher consequential and proactive level. In actuality, the event-oriented nature of the former may, in fact, trigger the proactive reasoning of the latter. In the context of homeland security, for example, such reasoning may produce: a warning\(^7\) that hazardous material is en route; a warning that a truck has not reached a waypoint within a certain time limit; an alert that a truck has not reached a waypoint within a more critical time limit; a warning that a truck is near a higher risk area; an alert that a truck has stopped for more than a certain time near a higher risk area; an alert that the loaded weight of a truck does not match the final weight at the border check point; and so on.

Within the same homeland security context (i.e., specifically inland border control), typical higher level agent inferencing capabilities may include warnings and alerts that a particular combination of circumstances involving encyclopedic data and truck-based or convoy-based confirmation data entered at waypoints and checkpoints constitutes a higher risk situation. Examples include, a particular driver transporting certain kinds of goods, or the combination of an authorized substitute driver taking an authorized alternative route without any apparent reason, and taking a significantly longer time between two consecutive checkpoints. While none of these individual anomalies might be sufficient to cause concern, their combined occurrence may well constitute a risk requiring further actions.

### 3.2.4 Benefits of the KMES® Approach

Considerable time and cost savings can be realized in the KMES® approach, without sacrificing quality. In fact, the quality of the software developed can increase due to both the extensive formal validation and verification process appropriate for a core capability as well as the informal validation and verification resulting from its repeated use in the field. The principal benefits of KMES®-based software systems are threefold: early delivery of usable decision-support tools; decreased software acquisition costs; and, higher quality products. The increasing focus on adaptive planning capabilities in the military environment has placed a premium on the rapid development and fielding of software with flexibility and intelligence. The Adaptive

---

\(^7\) Typically, agents will communicate with the user at different levels of urgency. For example, a warning may simply draw the user’s attention to some particular event or situation, while an alert signifies that the user’s focused attention is urgently required.
Planning Roadmap\textsuperscript{8} defines \textit{adaptive planning} as the capability to create and revise plans rapidly and systematically, as circumstances require. KMES\textsuperscript{®}-based software systems have the potential for meeting this challenge by virtue of the following inherent advantages:

- \textit{Rapid delivery} of meaningful capabilities, with the potential of achieving a first usable product installation within three to six months after the initiation of a software development project.
- \textit{Lower cost} due to the replacement of the normally prolonged software development period with a much shorter KMES\textsuperscript{®} integration period.
- \textit{Greater reliability} and quality due to exhaustive core-component verification and validation in conjunction with the maturity that comes from extensive in-field use.
- \textit{Interoperability} through component design based on standard protocols and a decoupled, multi-tiered framework.
- \textit{Flexibility} to extend functional capabilities through plug-and-play components and an open architecture.
- \textit{Multiple deployment options} including net-centric delivery alternatives of hosted-managed services.

3.3 ICDM: Philosophy and Principles

For the past two decades the CADRC Center and more recently CDM Technologies have pursued the design and development of agent-based decision-support systems. Throughout this journey the CADRC and CDM Technologies have relied on the Integrated Cooperative Decision-Making (ICDM) software development toolkit (Figure 3.3) to assist them in the creation and management of such systems.

Not only does ICDM function as an accelerator (i.e., rapid development) and stabilizer (i.e., built-in robustness and fault tolerance) in the development of decision-support systems, but it also provides a concrete vehicle for representing the key concepts and philosophies that the CADRC Center and CDM Technologies have found to be useful for the success of KMES-based systems (Pohl et. al. 2000, Pohl 1997). The key design principles on which ICDM is founded are collaboration-intensive, context-based representation, flexibility and adaptability, multi-tiered and multi-layered, within the framework of a service-oriented, distributable architecture (Figure 3.4).

Although no formalized reference model currently exists to concretely define a Service-Oriented Architecture (SOA) the software engineering community is in basic agreement as to the concepts associated with this approach (Newcomer and Lomow 2004). It is also generally acknowledged that under current circumstances not all of these concepts need to be comprehensively present in order to qualify as a SOA, thus highlighting the need for further standardization. Nonetheless, the following concepts are generally consistent among most SOA interpretations:

- \textit{Services:} An encapsulated body of behavior or capability that can be contracted by a requestor in an effort to satisfy some need.

• **Service Descriptions:** A detailed description of an available service, among other information, describing the input, output, and general contract for employment.

• **Advertising and Discovery:** Ability of a service to advertise its capabilities in an accessible manner (i.e., push or pull fashion) to prospective consumers.

![Figure 3.3: The ICDM development toolkit](image1)

![Figure 3.4: ICDM design principles](image2)

To fulfill their contracts, services may transparently employ other services thereby effectively obscuring the line between service providers and service consumers. Further, SOA-based systems lend themselves to flexible distribution across the available computing resources. This characteristic promotes considerably more flexible load-balancing and scaling capabilities than are offered by more monolithic architectures. In addition, because of their discrete nature, systems designed around a SOA methodology benefit from a low degree of coupling among their components (i.e., consumers depend on selected services and their associated contracts, not necessarily specific service instances or implementations). Due to this decoupled nature, capabilities within a SOA-based environment can be dynamically added, modified, or removed with little or no impact on service consumers as long as existing contracts are still supported and new or replacement services are discoverable through accessible registries.

### 3.3.1 Collaboration-Intensive

Certainly in the real world, collaboration among decision-makers and experts is a critical ingredient in making educated and effective decisions. This is especially true when operating across an extensive and varied set of domains. Through years of research in collaborative design the CADRC has found that this same quality extends to the realm of agent-based decision-support systems. Conceptually, the systems developed by the CADRC and CDM Technologies consist of dynamic collections of collaborators (both human and software-based) each playing a role in the collective analysis of a problem or situation and the consequential decision-making assistance required in formulating an accurate assessment and/or solution (Figure 3.5).
Whether human or software-based, collaboration within an ICDM-based system occurs in terms of a descriptive ontology (Chandrasekaran et. al., 1999). Until recently these ontologies were limited to describing information and knowledge that represents various aspects of the domain(s) over which the system is to operate. For example, in the domain of architectural design the applicable ontology would describe such notions as spaces, walls, accessibility, appropriate lighting, and so on. Although effective and certainly a fundamental element of an information-centric system a considerable portion of the system still remains in a form not necessarily supportive of highly collaborative environments. Further, these non-ontology based components require separate and dedicated interfaces along with specialized management. A number of the services that collaborators within an ICDM-based system interact with (e.g., time, query, subscription, execution, reasoning, etc.) were still presented as client-side adjunct-based interfaces requiring additional management to support collaboration. For example, if two clients wish to share or discuss the same subscription profile, a separate mechanism for identifying and referencing the collection of interests is required. In this case, the interface would be the client-side Application Programming Interface (API) maintained by the subscription service itself. Although certainly possible, supporting such specialized functionality requires the particular services (i.e., the subscription service in this case) to present and manage a specific API to expose or match global references. Although subtle in nature, complexity such as this can easily escalate when considering the high degree of collaboration inherent in multi-agent KMES® components.

This limitation may be overcome by extending an ontology to include behavioral aspects of the decision-support system in the form of object-based constructs representing the services within the KMES® component (i.e., the services themselves are represented in the collaborative ontology in the same manner as information and knowledge). The only difference is that these distributed and shareable objects offer behavior in addition to information. As a result collaborators are able to interact with these services through the same distributed object operations that they would perform on the information and knowledge objects. Any constraints
identified in the behavior are enforced by the standard ontology management facility. The 
operations that can be performed on these ontology-based objects consist of the basic creation,
deletion, and modification functionality. To support the aforementioned example in which two 
collaborators wish to reference and discuss aspects of the same subscription profile, the two 
collaborators would treat the profile in question as just another set of multi-faceted, shareable 
distributed objects. In other words, similar to the manner in which rich information models, or 
ontologies are used as a basis for collaboration this notion is extended to include interaction and 
collaboration across the services that constitute the system itself. The effect is essentially that 
interaction with, and collaboration across, information, and now behavior (e.g., services), is 
reduced to a basic set of object manipulation capabilities. In this sense, the fundamental subject 
of all operations, whether in the form of access or manipulation, is an object. The only difference 
is that some distributed, shareable objects offer information and some offer behavior. The client- 
side portion of the ontology replaces the need for specialized client-side functionality.

3.3.2 Context-Oriented Representation

Prior to the commencement of the TRANSWAY® project the CDM Technologies had developed 
several decision-support applications utilizing (under license) the Integrated Cooperative 
Decision Model (ICDM) development toolkit for multi-agent systems (Pohl and Myers 1992; 
Myers et al. 1993). An ICDM-based application is based on an information-centric premise, in 
the sense that it incorporates an internal information model of objects, their characteristics, and 
the relationships that associate these objects to each other and the functional capabilities of the 
application (Myers and Pohl, 1994; Pohl et al. 1992; Pohl K. 2002).

The term information-centric refers to the representation of information, as it is available to 
software modules, not to the way it is actually stored in a digital machine. This distinction 
between representation and storage is important, and relevant far beyond the realm of 
computers. When we write a note with a pencil on a sheet of paper, the content (i.e., meaning) of 
the note is unrelated to the storage device. A sheet of paper is designed to be a very efficient 
storage medium that can be easily stacked in sets of hundreds, filed in folders, folded into 
volumes, and so on. As such, representation can exist at varying levels of abstraction. The 
lowest level of representation considered in Figure 3.6 is wrapped data. Wrapped data consists of 
low-level data, for example a textual e-mail message that is placed inside some sort of an e-mail 
message object. While it could be argued that the e-mail message is thereby objectified it is clear 
that the only objectification resides in the shell that contains the data and not the e-mail content. 
The message is still in a data form offering a limited opportunity for interpretation by software 
components.

A higher level of representation endeavors to describe aspects of a domain as collections of inter-
related, constrained objects. This level of representation is commonly referred to as an 
information-centric ontology. At this level of representation context can begin to be captured and 
represented in a manner supportive of software-based reasoning. This level of representation 
(i.e., context) is by far the most empowering design principle on which ICDM is based. Further, 
as mentioned in the previous section portions of this context may be extended to exhibit 
behavior. In addition to services, however, distributed behavioral objects can also be employed 
as a mechanism for supporting the notion of facades.

Existing as one of the fundamental design patterns employed in object-oriented design, facades 
provide a level of derivation attained from the particular representation or ontology on which
they are based (Pohl K. 2001). In the case of ICDM and the kinds of ontologies it manages, facades offer a method of supporting and managing an alternative perspective from that modeled in the ontology from which they are derived. In other words, ICDM-based facades allow the perspective inherent in a particular model of a domain to be augmented, or in some way altered to support a more appropriate (i.e., to the façade user) representation of the concepts, notion, and entities over which that user is operating (Figure 3.7). Note that user in this sense refers to any accessing component. While certainly useful in systems supporting multiple perspectives caution must be employed in preventing abuse by introducing inconsistency and unnecessary duplication.

Facades can also be utilized to support real-time calculations. In this sense, the façade derivation would involve a calculation or algorithm perhaps based on one or more attributes of the base object(s). For example, consider an architectural space exhibiting length, width, and height described in American pound/foot units which is to be accessed by a design system that only understands Metric kilogram/meter units and also requires space volumes. Utilizing ontology-based facades a model could be developed in which, not only the length, width, and height, but also the volume of the space could be calculated and presented to the design system in terms of Metric units.

### 3.3.3 Extensible and Adaptive

One of ICDM’s primary goals is to support a high degree of flexibility in respect to the configuration of its components both at the development and execution levels. ICDM supports the addition, replacement, and reuse of software components in the context of agent-based, decision-support systems, and achieves this goal by reducing inter-component coupling to an absolute minimum (Figure 3.7). There are two key ICDM properties that permit this flexibility. First, all collaboration between clients takes place via, and in terms of the informational ontology (i.e., distributed objects). No direct communication exists between collaborators. The result is a
collaborative environment in which client identities are essentially irrelevant in respect to this process.

The second property deals with the manner in which clients access and interact with the ontology. ICDM offers a standard interface component known as the Object Management Layer (OML) which both shields accessors from the complexity of ontology management as well as provides an abstracted view of the ontology. Clients of OML interact with the ontology via object wrappers based on a set of corresponding ontology-specific templates. Promoting the notion of adaptability, these templates are discovered by OML as a runtime activity. The resulting support for dynamic definition permits elements of the ontology to be extended, eliminated, or even redefined during the course of a runtime session.

Apart from the ability to adapt to an evolving definition of a domain, adaptability is also supported in interaction with external systems. This level of adaptability functions in conjunction with the concept of façades mentioned earlier. Replacing the classical approach of building a dedicated and separate translation bridge between collaborating systems, ICDM promotes the incorporation of such translation into the ontology itself. In other words, using ICDM’s support for ontology-based facades, translation or derivation of each system’s perspective can be encapsulated and managed solely within façade objects. The resulting translation facility exists as a set of behavioral façade objects accessed and manipulated in a manner no different than is applied to other ontology objects. The result is an elegant design where support for translation-based communication between disparate systems is seamlessly incorporated as part of the ontology.

3.3.4 Multi-Tiered and Multi-Layered

The forth design principle to which ICDM adheres addresses the architectural organization of ICDM-based systems. More specifically, this principle identifies distinct separations between areas of functionality at both the conceptual (i.e., tier) level and the more concrete (i.e., layer) level. Conceptually, the architecture of an ICDM-based decision-support system is divided into three distinct tiers namely, information, logic, and presentation. To manage its particular domain each tier contains a number of logical layers that work in sequence (Figure 3.8).

As the name suggests the information tier houses both the information and knowledge (i.e., ontology) being operated on in addition to all of the mechanisms needed to support management, transport, and access. The information is further delineated into layers. The first of these is the Object Management Layer (OML) described in an earlier section. Below the OML resides the Object Access Layer (OAL) responsible for managing access to the Information Tier. The OAL exists as a level of abstraction below OML and interfaces directly with the Object Transport Layer (OTL). Based on the CORBA specification (Mowbray and Zahavi 1995) the OTL is responsible for communicating the various requests and subsequent replies for distributed information and behavior issued through the OAL throughout the system. The OTL is the only layer that forms a dependency on an underlying communication protocol. As such, support for alternative communication facilities can be implemented with minimal impact on either the OAL or the OML. This is an excellent example of the benefits of a layered architecture in supporting component reuse and replacement.

The Logic Tier contains the business rules (i.e., agents) and analysis facilities by which these rules are managed. Although extensible to include other forms of reasoning the current version
of ICDM focuses on opportunistic rule-based analysis. Regardless of which form of reasoning is employed this capability is supported by two layers namely, the Business Rule Layer (BRL) and the Business Engine Layer (BEL). The BRL is primarily system-specific and contains the agent-based analysis facilities resident in the system. Execution of agents is in turn managed by the BEL. To integrate the Logic Tier with the Information Tier the BEL interfaces with OML permitting the agents to both access and contribute to the ontology.

The final tier is the Presentation Tier. This tier is responsible for interfacing with the various users of the system. In this sense a user may be a human operator or an external system. In the case of a human operator support is provided through a Graphical User Interface Layer (GUIL) that presents and promotes interaction with the contents of the Information Tier. In the case of an external system, support takes the form of a Translation Layer (TL) that manages the mapping of representations between systems. Like the GUIL, access to and from the Information Tier is supported by OML.

3.4 The TRANSWAY® System Architecture

The TRANSWAY® suite of services is integrated within a distributed, three-tier, service-oriented architecture, designed and implemented according to the SOA and KMES® concepts and principles discussed in Sections 3.1 and 3.2. The development environment was provided by the ICDM framework and toolkit, which allowed a good portion (about 30%) of the source code on both the server and client sides to be generated automatically. In addition, existing KMES® components were utilized where available and only a relatively small number of new services were required to be developed. These new components were designed as generic services so that they could be reused as KMES® modules in future systems.

Figure 3.9 provides an illustration of the key components and services within each of the three tiers of the TRANSWAY® system (i.e., information, logic, and presentation tiers) and how they interact with one another. Components and services in the presentation and logic tiers may interact with components and services in the information tier but cannot interact with one another.

The information tier consists of an ontology, hereinafter referred to as the Domain Model, and various services and components that interact with it. The Domain Model serves as the internal information model that provides the necessary context for the intelligent capabilities of the TRANSWAY® system (e.g., route planning and the automatic generation of a distribution plan). It is in essence a virtual representation of the information involved in the application domain (i.e., the military distribution process domain) and consists of objects, their characteristics, and the relationships among those objects. The Domain Model is discussed in more detail in Chapter 4. The main services and components that interact with the Domain Model are the Facades, the Subscription Service, the Persistence/Query Service, and the Scenario Server. The Facades, Subscription Service, and Persistence/Query Service all assist with updating or receiving information from the Domain Model. The Scenario Server assists with starting and stopping services and components in the logic tier based on activity in the presentation tier. The Imagery Service is utilized by the presentation tier for retrieving map data. All of these services and components are discussed in more detail in the given order in subsequent chapters.
The logic tier consists of multiple agents and services that use the Tabu Search algorithm (Karaboga and Pham 2000, Glover and Laguna 1997) to find approximate solutions to the logistics planning and re-planning problem. These agents and services communicate with one another via the information tier in accordance with a data blackboard model and are discussed in more detail in Chapter 6.

The most important components of the presentation tier are the Thick Client and the Thin Client. The Thick Client, which is discussed in detail in Chapter 7, allows the user to interact with a georeferenced spatial framework (i.e., various maps), as well as reports, graphs, and alerts. Whereas the Thin Client discussed in Chapter 8, allows the user to interact with only a limited subset of functionality from the Thick Client through a web client interface. The presentation tier also
includes the Generic Space Generator (GSG) and the GSG Mapping service (which are both ICDM components) because they are the backbone for the Thick Client. GSG and GSG Mapping are discussed in Sections 7.2 and 7.3, respectively.
4. The Internal Information Model

An ontology can be characterized as an explicit specification of a conceptualization. For a software application, what exists is that which can be represented in the form of objects. This set of objects, and the describable relationships among them represents all the information and knowledge that can be known in the context of the applications that employ them. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms.

The TRANSWAY® ontology or Domain Model is an object-oriented representation of the military distribution problem domain and is rich in the objects, attributes, associations, and parent-child relationships, which govern the tasks and activities that are germane to this Community of Interest (COI). In this chapter, the term Domain Model refers to the Java classes that are utilized throughout all TRANSWAY® specific components. There are many other parts that come into play during the creation and utilization of the Domain Model. Figure 4.1 shows the Domain Model (i.e., Java classes), along with the initial Unified Modeling Language (UML) based model, the various ICDM transform scripts, the XML-based mapping, and the database (defined by a Data Description Language (DDL) file).

Creation of the Domain Model began with the design of a UML-based model, based on the following guidelines. The Domain Model should be:

- **Simple:** Only necessary objects, attributes, associations, and parent-child relationships should exist. This is required for performance purposes (e.g., queries should not have to contain unnecessary associations/joins) and for code maintenance purposes (e.g., unnecessary casting, association traversal, and attribute maintenance should not have to consume developer time).
- **Intuitive:** Should be very close to the actual real world problem domain. For example, if the concept of a Supply Center is required to be represented then a SupplyCenter object should exist in the Domain Model.
• **Efficient:** – If the size of a certain association is expected to be very large, then a one-way association or an `sObjectReference` (i.e., a reference to an object) is used. This approach is again preferred for reasons of performance. For example, if an association is expected to grow to size 5000 and it is stored as a `Set`, then due to the nature of a `Set`, each time an object is added to the `Set`, the `Set` must be pulled into local cache and searched to assure duplicates are not made. Since this can be very time consuming, it was decided to utilize one-way associations and `sObjectReferences` instead.

ICDM transform scripts were performed on the UML-based model in order to produce the actual Domain Model (i.e., Java Classes), the XML-based mapping, and DDL file. The Domain Model classes were made available to the development team via a jar file. The DDL file is used to create the database tables that store TRANSWAY® data. Finally, the XML-based mapping is utilized during execution by the Persistence/Query Service discussed in Chapter 5. The TRANSWAY® Domain Model is divided into 10 interrelated domains, discussed separately in the following sections.

4.1 The Main Domain

The Main or parent domain, shown in Figure 4.2, contains metadata objects and objects from which all other TRANSWAY® domain objects are derived. The Main domain contains the primary `TranswayObject`, the `TranswayType` object, a `ClassInformation` object, and an `AttributeInformation` object. The `TranswayObject` and the `TranswayType` object are abstract in nature. Every other object in the TRANSWAY® domain inherits from one of these two abstract objects. The `ClassInformation` and `AttributeInformation` objects contain class and attribute metadata and are initialized upon the creation of any particular distribution plan generation sequence.

![Figure 4.2: The Main Domain](image-url)
4.2 The *Time* Domain

The *Time* domain, shown in Figure 4.3, contains objects that allow the representation of time in TRANSWAY®. The domain contains a *Clock* object and a *TimeWindow* object. The *Clock* object is a singleton object that represents the current system time, which may be different from the real world time. While several *TimeWindow* objects can exist concurrently, they can only exist in the context of some other object. For example, an *Impediment* object in the *Physical* domain can have a *TimeWindow* representing its window of activity.

![Figure 4.3: The Time Domain](image)

4.3 The *Types* Domain

The *Types* domain, shown in Figure 4.4, contains objects that allow the representation of type data or encyclopedic data such as kinds of *Persons*, *Organizations*, *Containers*, *Conveyances*, *ConveyanceLayouts*, *Cargo*, *Alerts*, and *Agents*. All type data are loaded during the initialization of the system. Other objects in TRANSWAY® can associate to a particular type object for classification purposes. For example, a single *CargoType* object can exist for the type of cargo with National Stock Number (NSN) 5310006842725 and many *Cargo* and/or *Requirement* objects can associate to that type object. Type objects allow a developer to easily query for all objects of a particular type.
Figure 4.4: The *Types* Domain
4.4 The Constraints Domain

The Constraints domain, shown in Figure 4.5, contains objects that allow the representation of various constraints in TRANSWAY®. The domain contains an EmbeddedConstraint object, an ObjectiveConstraint object, a PropertyConstraint object, a PersonConstraint object, and an ItemConstraint object. Most of the constraint objects are created along with type data on the creation of a new scenario. For example, EmbeddedConstraint objects are created as violatedConstraint associations of AlertTypes, and PropertyConstraint objects are created as constraints with a value and comparator for PersonTypes, all ItemTypes, and ConveyanceLayout types. It is beneficial for constraints to be associated with type data because future constraint objects can be added to TRANSWAY® without any major impact on the existing code.

Figure 4.5: The Constraints Domain

4.5 The Items Domain

The Items domain, shown in Figure 4.6, contains objects that allow the representation of conveyances and cargo. This domain consists of Item objects such as the Conveyance object, the Container object, and the RolledUpCargo object. Conveyance objects can be created in the context of owning OrganizationEntities. They have a defining ConveyanceType and can be defined as unavailable by a set of TimeWindows. Many Container and RolledUpCargo object pairs can be created in TRANSWAY® and associated with some location such as, for example, a SupplyCenter. The Conveyance objects and the Container-RolledUpCargo object pairs are necessary in support of the automated planning and re-planning tasks performed in TRANSWAY® by the agents.
Figure 4.6: The *Items* Domain

Figure 4.7: The *Environment* Domain
4.6 The Environment Domain

The Environment domain, shown in Figure 4.7, contains objects that allow the representation of the physical environment. This domain contains a Waypoint object, a SupplyCenter object, a RouteSegment object, a Note object, an AreaToIgnore object, an AreaOfInterest object, and Impediment objects. With the exception of the abstract Impediment object, all of these objects can be placed into a geospatial context such as a geo-referenced map. The most notable of these objects are the SupplyCenter and RouteSegment objects. A SupplyCenter object can contain Cargo that is to be delivered and can be the home to Organizations that have Requirements. The availability or unavailability of a SupplyCenter object is represented by association with one or more TimeWindows. RouteSegment objects interconnect SupplyCenter objects to represent the paths for delivering Containers with Conveyances.

4.7 The Organization Domain

The Organizations domain, shown in Figure 4.8, contains objects that allow the representation of organizations and personnel. This domain contains OrganizationEntity objects (i.e., an Organization object, a Person object, and a Detachment object). All OrganizationEntity objects can be created in TRANSWAY® and are represented in a tree like structure. For example, an Organization object may have associations to many Person objects and many child Organization objects, and the child Organizations can have associations to many Person objects and other child Organizations, and so on. There are many other important associations, for instance, an OrganizationEntity object can be associated to many Item objects and Person objects can be associated to many TimeWindow objects that represent a schedule.

Figure 4.8: The Organization Domain
4.8 The Requirements Domain

The Requirements domain, shown in Figure 4.9, contains objects that allow the representation of requirements or requests. The domain contains a Requirement object and SubRequirement objects (i.e., an OrganizationalRequirement object and a CargoRequirement object). Requirement and SubRequirement objects are created in TRANSWAY® in the context of a requesting Organization. Planning and re-planning activities by agents are based on their contents. The CargoRequirement objects are associated with a CargoType, and OrganizationalRequirement objects are associated with an OrganizationEntity.

![Figure 4.9: The Requirements Domain](image)

![Figure 4.10: The Alerts Domain](image)
4.9 The Alerts Domain

The Alerts domain, shown in Figure 4.10, contains objects that allow the representation of agent alerts. Many Alert objects can exist in TRANSWAY® and each represents an alert given by an agent. Each Alert has an AlertType by which it is categorized.

4.10 The Planning Domain

The Planning domain, shown in Figure 4.11, contains objects that allow the representation of plans. The domain contains a Transfer object, a CargoTransfer object, and a Solution object. When a plan is generated in TRANSWAY®, it consists of multiple Solution, Transfer, and CargoTransfer objects. Each Transfer object represents one leg of a plan. In order to accommodate the large amount of creations, modifications, and deletions of Transfer objects, most associations have been replaced with a sObjectReference (i.e., a reference to an object) or an array of sObjectReferences.
5. Information Services and Components

TRANSWAY® utilizes various ICDM and KMES® specific information services and components. Many of these utilized services and components act as mediators for the Domain Model (discussed in Chapter 4) and other TRANSWAY® components (e.g., the Thick Client discussed in Chapter 7). These mediating components and services, all of which assist with updating and/or receiving information from the Domain Model, are the Facades, Subscription Service, and Persistence/Query Service. The other major information services and components that are utilized by TRANSWAY® are the Scenario Server and the Imagery Service, discussed below in Sections 5.4 and 5.5, respectively. The Scenario Server assists with starting and stopping agents based on activity in the Thick Client and the Thin Client (discussed in Chapter 8). The Imagery Service is utilized by the Thick Client for retrieving map data.

5.1 Facades

The Facades are an object-oriented design pattern providing a unified interface to represent objects in the Domain Model and hence the database. Facades offer a higher level interface than the Domain Model and are easy to maintain in the Java environment. From a general point of view Facades are used to facilitate:

- **Readability:** By making the software library easy to use and understandable.
- **Scalability:** By forcing Facades to follow strict guidelines supporting maintenance and reducing outside dependencies.
- **Portability:** By allowing use across multiple domain and report interfaces.

In TRANSWAY®, Facades act as a mediation layer between the Domain Model and the presentation tier. This allows quick and easy access to the database for both data analysis and data manipulation. A Facade is essentially a Java Bean and a Data Accessor Object. The Java Bean consists of mutator and accessor properties corresponding to the properties of the Domain Model object it represents. The main benefit of using Java Beans is portability across multiple domain and report interfaces. Java Beans encapsulate data about an object in the Domain Model (and hence the database), however, they have no direct access to the Domain Model (or database) itself.

To facilitate this behavior, each Java Bean is paired with a Data Accessor Object (DAO). The DAO handles the transactions between the Java Bean and the Domain Model. DAOs also handle synchronization with the Domain Model. Each time a Java Bean is modified, the DAO propagates the updated information from the Java Bean to the Domain Model (and hence the database). Each time the database object is written to, the DAO also refreshes the Java Bean based on the new values. This allows concurrent modification of database objects from multiple users on the same domain application.

In addition to a Java Bean and a DAO, each Facade class includes a Delegate. The Delegate is a collection of static methods that perform predetermined queries for information contained in the database. They provide convenient methods for common tasks such as fetching Facades and locating associations between objects in the Domain Model. For example, the Supply Center Delegate incorporates a method to retrieve all Supplies at a particular Supply Center location.
5.2 Subscription Service

The Subscription Service supports publish and subscribe capabilities that enable clients (e.g., Thick Client and Thin Client) to subscribe to Domain Model interests based on class name, object ID, or property. The service is integrated with the domain classes to allow for notification of create, property change, and delete events. During a transaction, events are sent to a queue whenever a class is created or deleted and whenever a property is changed. The queued events are then published once the transaction commits. Satisfaction of interest criteria results in a notification to the subscriber.

The ICDM Subscription Service interface (i.e., API) supports the registration of listeners with class-based or object-based interests for creation and deletion events, as well as property change events. The API supports event-driven programming for handling changes in the shared domain. This is useful for data synchronization. For example, the Facade DAOs register the listeners that have an interest in property change events. Therefore, when property change events that match these interests are published, the event listener will handle the refreshing of the Facade Java Bean. In other cases, events are used to trigger processing and react to changes. For example, the planning agent listens for new requirements, route changes, and supply changes so that it can re-plan accordingly.

The Subscription Service is supported by the ActiveMQ Java Messaging Service (JMS) broker. ActiveMQ is a leading Open Source message broker. It was selected on the grounds of its performance and robust capabilities, particularly in respect to clustering.

For each class in the Domain Model, a separate JMS topic is defined for each type of event (i.e., creation, deletion, and property change) and a topic for each type of event is defined for the Domain Model in general. Class interest listeners are registered on a subscriber to a class topic based on event type, while object interest listeners are registered on a subscriber to domain topic based on event type. If property constraints are specified for a subscription, a JMS message selector is created for the registered interest listener. Messages are published to the class topic for the object that the event occurred on and, for inheritance support, to the parent class topics. The messages are then published to the topic subscribers for which no property constraints have been specified. Additionally, all message selectors that have been registered for the particular topic are evaluated and messages are published to those topic subscribers whose property interests are satisfied.

5.3 Persistence/Query Service

TRANSWAY® makes use of Hibernate®, the Oracle Relational Database Management System, and the Oracle thin JDBC driver for its persistence and query capabilities. In addition, ICDM provides tools for generating Hibernate compatible classes with integrated support for association management and event notification, and for generating the object-relational mapping. Both the classes and the object-relational mapping are generated from XMI derived from a UML class diagram representing the Domain Model (discussed in Chapter 4). Oracle is a leading database for decision-support and on-line transaction processing. It was selected because of the

---

existing license agreement between USTRANSCOM and Oracle Corporation and because it is a robust, high performance database.

Hibernate is a leading solution for object-relational mapping (ORM) and object persistence in general for the Java platform. It provides a bridge between the object-oriented world and the relational world, making it possible to have both a finely-grained and richly typed Domain Model and to make use of the database for powerful and efficient queries. It was selected on the grounds of performance and its functional capabilities. In particular, it supports lazy loading eager fetching for efficient loading of collections and associated objects, and optimistic locking for concurrency control to maintain data integrity. Furthermore, it offers an API for supporting transactions in an unmanaged environment, is compatible with most (if not all) leading open source and commercial databases, and provides a tool for generating a database schema from the mapping.

The object-relational mapping is based on one table per hierarchy inheritance mapping. That is, the mapping domain is partitioned such that each hierarchy maps to a course grained database entity. This strategy supports polymorphism for all types of associations and is therefore well positioned for performance optimization, since it requires fewer SQL joins.

The persistence and query logic has been designed and implemented for optimum performance and to handle concurrency for supporting multiple on-line users. For performance, queries are optimized to minimize database calls. For handling concurrency, stale object exceptions are minimized by constraining the session scope to a particular use-case or application level transaction. In addition, optimistic locking is favored in place of pessimistic locking to improve performance and minimize lock contention.

5.4 Scenario Server

The Scenario Server is responsible for starting and stopping the agents for any given planning or re-planning sequence when a user is logged into the TRANSWAY® Thick Client. The sequence diagram in Figure 5.1 shows the general interaction between the Thick Client, the Scenario Server, the Subscription Service, and the agents. When the Scenario Server starts up, it starts the Subscription Service Broker and the RMI Registry of that agent. These services are used to support agent communications with each other and the database.

When a user logs into the Thick Client, a message is sent to the scenario server directing it to initialize the scenario support services. The Scenario Server will register the scenario with the Subscription Service Broker, which will provide the messaging capabilities between every client registered to that scenario. After the scenario is registered with the broker, the Scenario Server will start up the Planning Agent, Impediment Agent, Feasibility Agent, and Truth Maintenance Agent. Upon initialization, the agents register themselves with the Agent RMI Registry. When the Planning Agent needs one of the services provided by the other agents, the Planning Agent will get the RMI interface for that agent, from the agent registry. When the last user logging out of the scenario, the server will begin shutting down the agents and un-register the scenario from the Subscription Service Broker. The Scenario Server handles agent fault tolerance as well as the asynchronously starting and stopping of multiple scenarios.
5.5 Imagery Service

The Imagery Service provides efficient Internet access to high resolution geospatial imagery. It is a RESTful web service\(^\text{10}\) supporting token-based authentication, which provides access to

\(^{10}\) Representational State Transfer (REST) is a software architecture for distributed hypermedia systems such as the World Wide Web. Systems that follow REST principles are commonly referred to as RESTful.
imagery tiles by layer, row, column, and zoom level. If no layer is specified in a query, the imagery service is configured to returned tiles from the primary combined layer, which combines the best quality imagery at each level to provide a greater overall coverage.

Representational State Transfer (REST) defines a stateless client-server architecture in which web services are viewed as resources represented by their URLs. It is known for its scalability, performance, security, reliability, and extensibility, and it reduces complexity. Because it conforms to the W3C Web Architecture, it requires only existing web technologies.

The high level of performance of the imagery service is possible due to the fact that the imagery is pre-rendered into tiled image pyramids. The benefit of this is twofold, it requires an application to only download the tiles for the area being viewed and secondly the tiles are immediately available since they are pre-rendered.

The image pyramid, shown in Figure 5.2, is based on the power of two, where each level has increasingly higher resolution (i.e., at a higher level each pixel covers a smaller area). Each level has $2^n$ rows and $2^{n+1}$ columns, since there are twice as many degrees longitude as there are degrees latitude. The tiles are 512 x 512 pixels and each successive level is twice the resolution of the previous level (e.g., a tile at level 0 corresponds to four tiles at level 1).

![Figure 5.2: The image pyramid](image.png)

Image pyramids were built for Blue Marble Next Generation (BMNG), Earthsat Naturalvue, and military Compressed Image Base (CIB) imagery. To assist in this process the Pyramid Builder tiling utility was developed to make use of MapServer\footnote{MapServer is an Open Source Internet map server that was developed by the University of Minnesota.} for generating the images, together with a quadtree spatial index to look forward and skip areas with no coverage. MapServer is well suited for the task as it handles the mosaicing, re-sampling, edge alignment, and merging of pixels from multiple images, and it is built on Geospatial Data Abstraction Library (GDAL), which is a translator library that supports the raster geospatial data formats for all the imagery tiled (i.e., National Imagery Transmission Format (NITF), JPEG2000, and Geographic Tagged Image File Format (GEOTIFF)). Prior to running the tiling utility, the data was indexed and
MapServer was configured to access the data and serve true-color Portable Network Graphics (PNG) images. Indexing the data involved creating a *shapefile* tile index and a *quadtree* spatial index for each of the data sets using the `gdaltindex` and `shptree` utilities packaged with the FWTools\(^\text{12}\) Geographic Information System (GIS) distribution.

\(^{12}\) FWTools is an Open Source GIS Binary Kit for the Windows and Linux operating systems.
6. The TRANSWAY® Agents

The TRANSWAY® agents provide automated planning and re-planning capabilities for both static and dynamic scenarios. They are built around the Tabu Search strategy, which is well suited for searching combinatorial solution spaces where a certain combination of atomic entities is considered a solution (Glover and Laguna 1997). This chapter is essentially divided into three parts. Section 6.1 reviews Tabu Search approaches from a more general viewpoint. Sections 6.2 to 6.7 describe the design and implementation of the particular Tabu Search strategy employed in the TRANSWAY® system, and Section 6.8 discusses related work by others in the military field. The material presented in Sections 6.2 and 6.8 is based on graduate research conducted as part of a Master degree thesis in the Computer Science Department at Cal Poly, San Luis Obispo13.

6.1 The Tabu Search Algorithm Approach

Tabu Search is a metaheuristic that can be superimposed on other procedures to prevent them from becoming trapped in local optima (Glover 1990). The search algorithm attempts to direct heuristics toward an optimal solution through the use of memory structures (Glover 1996). Memory allows Tabu Search to greedily explore the search space without becoming trapped in local optima and effectively traverse through the most promising areas of the search space. Tabu Search quickly converges to a sub-optimal solution, while continuously improving the best-known solution. The Tabu Search algorithm has been applied to several combinatorial optimization problems (Colletti and Barnes 2004, Nanry and Barnes 2000, Rego and Roucairol 1995).

Tabu Search is a local search technique that starts with an initial solution and applies moves to transform the initial solution into new solution states. The feasible moves for a solution are known as the move neighborhood and encompass the possible trajectories of the search. An objective function determines the attractiveness of moves with respect to the current solution. Examples of moves include the changing of values assigned to variables, adding or deleting an element from a set, or swapping the positions of elements in a set (Glover 1990). Once the selected move has been applied to the solution, the search process repeats starting from the new solution state. Memory is used to prevent cycles in the search. The search continues until user defined criteria are met.

Tabu Search must consider a tradeoff between intensification and diversification techniques (Blum and Roli 2003). Intensification techniques attempt to improve the current solution by only considering moves within the current area of interest. These techniques allow the search to improve the current solution, but trap the search in local optima. Diversification techniques force the search to explore new areas of the search space, but degrade the solution. However, diversification techniques can lead the search into improved areas of the search space. A useful implementation of Tabu Search must utilize both techniques to effectively traverse through the search space. Consequently, Tabu Search alternates between intensification and diversification through the use of memory structures and candidate lists (Glover and Laguna 1997).

---

13 Weber B. (2006); ‘Optimization of Strategic and Operational Planning Utilizing Tabu Search’; Master degree thesis, Department of Computer Science, California Polytechnic State University (Cal Poly), San Luis Obispo, California, June.
6.1.1 Tabu Search Memory Structures

The core of Tabu Search is embedded in its short-term memory process (Glover 1990). However, it is necessary to fully exploit all aspects of memory to implement an effective Tabu Search strategy (Glover 1996). The two forms of Tabu Search memory are short-term memory and long-term memory. Short-term memory is used to escape from local optima, while long-term memory prevents re-cycling during the search process.

A purely greedy hill climbing algorithm will improve the solution until a local optimum is reached, causing the algorithm to become trapped. Tabu Search adds short-term memory to the hill climbing approach to allow the search to escape local optima and traverse through humps in the search space (Glover 1990). The short-term memory stores the moves recently applied to the solution and causes these moves to become taboo. The taboo status of a move forces the search to explore different areas of the search space.

Some problems have been solved by versions of Tabu Search utilizing only short-term memory (Glover 1990). However, several difficulties may arise when only short-term memory is implemented. The size of the short-term memory must be large enough to allow the search to escape a local optimum. Otherwise, the search can become trapped in an area of the search space. However, using an excessively large short-term memory may prevent the search from exploring promising areas of the search space. Short-term memory allows the search to escape from local optima, but this does not imply that the search will eventually reach the optimal solution. The search may cycle between two local optima, never exploring additional areas of the search space.

Long-term memory overcomes many of the shortfalls associated with short-term memory. It tracks the solution states that have been visited, as well as the moves that have been applied to these solution states. Ideally, the size of the long-term memory should be infinitely large. Long-term memory prevents re-cycling by restricting the search from selecting moves that have already been applied to the current solution state. Memory also operates as a basis of strategies for intensifying and diversifying the search process (Glover 1990). Long-term memory allows the search to revisit solution states, but forces the search to select different moves in these states.

Long-term memory is used to guide the search process through the use of upper level strategies, such as strategic oscillation (Glover and Laguna 1997). Strategic oscillation forces the search to alternate between intensification and diversification modes. The goal of long-term memory is to allow the search to exploit areas of the search space and find near-optimal solutions, while converging toward a global optimum solution. The amount of time spent in different modes depends on the application domain. If the purpose of the search is to quickly determine a good enough solution, then the majority of the search time should be spent using intensification techniques. If timing is less critical for the search, then allowing the search to focus on diversification techniques may result in the discovery of higher quality solutions.

6.1.2 Reactive Tabu Search

Initial versions of Tabu Search used fixed size short-term memories, traditionally with a size of five to twelve (Glover 1990). The size of the short-term memory is known as the tenure. Using a fixed tenure prevents the search from effectively solving a wide range of problems, which results from the tenure being tuned to the training set problems. A solution to this problem is reactive short-term memory (Battiti and Tecchiolli 1994), which adjusts the tenure based on the current
performance of the search. When the search finds improved solutions, the tenure is decreased and this allows the search to fully exploit the current area of the search space. When the search does not find improved solutions, the tenure is increased. Increasing the tenure causes the search to explore new areas of the search space. Using a reactive tenure allows the short-term memory to guide the search between intensification and diversification techniques. Reactive short-term memory allows Tabu Search to more effectively explore the search space, but introduces additional parameters into the search algorithm. Therefore, an algorithm may perform well on a training set of problems, but perform poorly on a test set of problems. One solution is to use feedback to achieve a self-tuning algorithm (Battiti 1996). Feedback adjusts the search parameters based on the performance of the search and can therefore be used to modify both the size of the short-term memory and the focus of the long-term memory. For instance, if the search discovers improved solutions, then the search should use intensification techniques. However, strategic oscillation may force the search to start using diversification techniques, moving the search into a new area of the search space. Feedback can be used to override strategic oscillation when the search finds good solutions, thereby allowing the search to continue exploiting a promising area of the search space.

6.1.3 Tabu Search for Constrained Problems

Tabu Search has been applied to problems that place constraints on the feasible solutions of a problem. For instance, Tabu Search has been applied to a constrained version of the minimum spanning tree problem (Glover 1990) and has also solved constrained instances of the vehicle routing problem (Laporte et al. 2000). Constraints may render a problem more complex, as demonstrated by the minimum spanning tree problem examined by Glover (1990). However, a problem with many constraints may be solved using constraint satisfaction techniques (Barták 1999).

There are two approaches for solving constrained problems using Tabu Search: allowing only feasible solutions (Taylor et al. 2006); and, allowing infeasible solutions in addition to feasible solutions (Glover 1990). Feasible solutions are solutions that do not violate constraints, while infeasible solutions violate constraints. Maintaining a feasible solution ensures that every discovered solution will not violate constraints. However, enforcing a feasible solution prevents a search from considering a portion of the move neighborhood. Searches allowing infeasible solutions may discover superior solutions, but the discovered solutions may violate constraints. Penalties are assigned to moves that cause the solution to violate constraints, thereby forcing the search to consider only infeasible solutions when there are no good alternatives. The type of search that should be implemented depends on the structure of the problem being solved.

It is possible to use a form of Tabu Search that starts with an initially empty solution and adds elements to the solution as the search progresses. The search maintains a feasible solution by considering only moves that will not violate constraints. A search using this approach is known as a constructive search (Randall 2002), while a search that maintains a goal state is known as a solution-based search. Constructive searches represent the search space differently than solution-based searches, because every state does not represent a goal state. Therefore, the search space for a constructive search is larger than the search space for a solution-based approach. Constructive versions of Tabu Search have a different representation of the move neighborhood, which may improve or degrade the performance of the search.
6.1.4 OpenTS Framework

OpenTS is an object-oriented framework for the implementation of Tabu Search algorithms (Harder 2001). The framework provides an overview of the procedures required to implement a Tabu Search algorithm. The search begins with an initial solution, which is defined by the implementation. The move manager constructs feasible moves to transform the current solution. The objective function evaluates the objective value of each move with respect to the current solution. The Tabu list determines the taboo status of each move. The move with the best objective value that does not have taboo status is applied to the current solution. When an iteration has been completed, the new solution is set as the current solution and the search process repeats.

**Move Manager:** The move manager is responsible for generating moves for the current solution. A move represents a transformation to the current solution. The possible moves for a solution are defined by the move neighborhood. The move manager may generate all moves for the current move neighborhood or a subset of the moves. If the move neighborhood is large, then generating all possible moves is computationally expensive. The move manager is responsible for determining which moves are worth exploring. The subset of moves generated by the move manager is known as the candidate list (Glover 1990). Using candidate lists may prevent the search from discovering the optimal solution, but results in faster iterations.

From a generally point of view there are two approaches for creating candidate lists: randomization; and, strategic oscillation (Glover and Laguna 1997). A search can randomly select moves from the move neighborhood to construct the candidate list. Using randomness is simple and may force the search to examine influential moves that are not considered by more advanced techniques. However, randomness results in unpredictable search behavior. Strategic oscillation constructs the candidate list based on the current focus of the search, which alternates between intensification and diversification. Therefore, the move manager must determine which moves will intensify the search and which moves will diversify the search.

**Objective Function:** The objective function determines the objective value of a move, which in turn specifies the quality of the solution that results from the application of the move to the current solution. The objective function is not an intelligent component, but often requires the largest amount of computation. Candidate lists speed up the search by limiting the number of moves that are evaluated by the objective function. Another technique is to distribute this task to multiple processors or multiple machines in a distributed environment (Harder 2001).

**Tabu List:** The Tabu list determines the taboo status of moves. A Tabu list can be implemented using several memory structures, including short-term and long-term memory (Glover 1996). Short-term memory is implemented as a list consisting of moves that have been recently applied to the solution. Moves in this list are considered taboo until removed from the list. Long-term memory stores the visited solution states as well as the moves applied to the different solution states. The purpose of long-term memory is to allow solution states to be revisited, while preventing re-cycling in the search process.

**Solution:** The solution component represents a solution to the problem being solved. Since a problem may have several possible representations, it is important to select a
representation that allows the search to efficiently traverse through the search space. Using a heuristic to generate an initial solution for a Tabu Search sequence may improve the performance of the search (Harder 2000). Jump searches allow the search to start from a superior solution state, as compared to starting from an arbitrary solution. However, there are trade-offs involved with the jump search technique. The initial quality of the solution discovered by Tabu Search may be improved over a random solution, however, the jump search may start the Tabu Search sequence in a disadvantageous area of the search space.

6.2 The Agent Planning Approach in TRANSWAY®

The TRANSWAY® agents are required to have access to a wide array of knowledge, such as information about: conveyances; cargo; demands; location; and, routes. Each of the various types of knowledge contains constraints that define a solution space. For example, conveyances all have a maximum range. Therefore, a solution should not include a mission that allows a conveyance to travel further than its maximum range.

In the TRANSWAY® system this knowledge is represented by the Domain Model (discussed in Chapter 4) and the associated constraints. The Domain Model contains a set of concepts with associations representing the problem domain. Examples of these concepts include Requirement, Organization, Transfer, Supply Center, and Route. Associations between the concepts define how these concepts relate to each other, and give meaning to each concept in the model. As part of the knowledge representation, the constraints associated with the Domain Model contribute by defining the solution space of a problem instance. Examples of such constraints include: the impediment constraint; the time constraint; and, the vehicle capacity constraint. While some constraints can be represented as the properties of a concept in the Domain Model (e.g., a Vehicle has a maximum weight capacity), others need to be represented and maintained as agent logic (e.g., a vehicle needs to return to its home base after its mission has been completed).

Since the TRANSWAY® system incorporates multiple agents that perform specific computational tasks, threading and synchronization require particular attention. Often some of these computational tasks need to be performed in parallel or, more accurately stated, cannot be performed serially. An example of this requirement for concurrency is the need for one agent to monitor the current demand for supplies, while another agent continually calculates the all-pairs shortest path algorithm.

The problem that the TRANSWAY® agents tackle is a dynamic vehicle routing and scheduling problem. Much like the well known Traveling Salesman problem, the vehicle routing and scheduling problem falls into a class of problems that are NP-hard. This means that these problems grow in complexity quite fast, and it is impractical to try and examine every possible solution to a sizable scenario.

6.2.1 Separation of Trip and Plan Generation

The literature describes many different approaches to combinatorial problems of the type encountered in vehicle routing (Talbi 2002). Based on a review of this literature it was decided early on in the design of the TRANSWAY® agents to treat trip generation and scheduling as separate problems. It was noted that most of the approaches cited in the literature utilize not one but several strategies for solving the combinatorial problem. In solving NP-hard problems, while
the different strategies are normally domain specific, the commonality that appears to exist among most of the approaches tends to limit the search space of the problem by taking advantage of the known constraints of the system. This criterion was adopted as an important design feature of the Planning Agent, to limit the number of trips produced so that the combinations of trips that make up a better (i.e., more optimal) plan can be found more quickly.

6.2.2 Selection of Search Methodology

With the separation of trip and plan generation the planning part becomes primarily a search problem. As new trips are generated they need to be considered as possible components of a recommended plan. However, even with the limitation of the search space through the application of constraints, the combination of generated trips into valid plans is likely to be time consuming. The main consideration in selecting a search strategy was the ability to generate a partial solution first and then continuously improve it as the search continues. Several different search methodologies were considered, as follows:

• **Simulated Annealing**: This method is essentially a simulation of the annealing process in metals. A temperature value that simulates a cooling effect much like annealing is defined. This value eventually becomes cold enough to force the search to find a near optimal solution.

• **Genetic Algorithms**: This method involves breeding solutions and applying random mutations to evolve a population of best fit solutions.

• **Constraint Logic Programming**: This method involves using a search algorithm with discrete domains to find values that satisfy the given constraints (e.g., backward chaining).

• **Tabu Search**: This method is based on the concept that new solutions should not revisit portions of the solution space previously considered.

The Tabu Search method was selected because it is particularly suitable for the type of vehicle routing and scheduling problem encountered by TRANSWAY® (Crino 2002). Some desirable features of the Tabu Search method include:

1. **Incremental optimization**. A search can start with any possible solution of a problem instance or even an empty solution and work toward improving the solution iteratively (Figure 6.1).

2. **The use of a Tabu List**. A Tabu List prevents a search process from being dead-ended in a local space that the search has visited. Therefore the search is able to continuously explore some new potential solution neighborhoods.

3. **It looks for a ‘good-enough’ solution**. It quickly discovers a satisfactory solution, and if given more time, can find a better solution leading perhaps eventually to an optimum solution.

4. **Ability to adapt to changes**. Tabu Search is an iterative optimization process, which works well not only with static input but also with dynamic input. During iteration, when the search detects that the problem input has changed, it adjusts the current solution to be consistent with the new constraints and then continues to improve it in the same way as it did before.
In order to adapt the Tabu Search methodology to the specific requirements of the TRANSWAY® system there was a need to translate the mathematical representation of the Tabu Search algorithm into the object-oriented environment of the ontology-based system architecture. For example, the Mission object in the Domain Model, which is equivalent to the concept of trip used by Crino (2002), contains a reference to a conveyance object and a list of transfers representing each journey that the conveyance will embark on, together with its associated cargo.

Another theoretical notion that required translation was the concept of a move (Crino 2002). A move is typically defined as replacing one mission in a solution with another mission. However, a mission cannot be replaced by just any other mission. Crino (2002) uses the conveyance as a convenient identifier, so that one mission can be replaced by another mission if they share the same conveyance. This is not acceptable in the case of TRANSWAY®, because conveyances should be able to perform more than one mission. Therefore, in TRANSWAY® missions are identified by the degree to which the demand for supplies is satisfied. Accordingly, a set of missions can be replaced by another set of missions that satisfies all or a subset of the demands.

6.3 Overall Architecture

The agents in the TRANSWAY® system utilize multiple ICDM components and services in order to access the Domain Model (and the database) efficiently. Figure 6.2 shows the overall TRANSWAY® system architecture with an emphasis on the agents (i.e., the Planning Agent, the Feasibility Agent, the Truth Maintenance Agent, and the Impediment Agent). Components and services that are not pertinent to the agents have been grayed out and extra arrows have been added to show agent specific dependencies and component interactions.

In order to interact efficiently with the Domain Model, all of the agents utilize the Subscription Service and the Persistence/Query Service (see Sections 5.2 and 5.3). The Subscription Service allows components within the agents to register for creations, deletions, and modifications to the Domain Model. For example, when a user creates a Supply Center in one client, any other client that is registered for these creations will receive a notification. The Persistence/Query Service allows agent components to persist and query for objects in the Domain Model. This allows for agents to shape transactions and queries without having to access the database directly.

Figure 6.1: Iteration of the Tabu Search method (Harder 2007)
Among the agents shown in Figure 6.2, the Planning Agent plays the central role of implementing the Tabu Search strategy to provide planning and re-planning capabilities for the TRANSWAY® system. Other agents, namely the Feasibility Agent, Truth Maintenance Agent, and Impediment Agent, provide services to assist the Planning Agent and users to find quality solutions. For example, the Impediment Agent monitors the states of routes and reports for the Planning Agent if some of the routes are impeded, and the Feasibility Agent provides feedback to the user explaining why the Planning Agent is unable to satisfy requirements in a scenario.

Although the agents are designed and implemented as independent software components, they collaborate concurrently to achieve the common objective of assisting users to find quality solutions as quickly as possible for both static and dynamic scenarios. The Impediment Agent
and the Planning Agent communicate with each other directly through JMS (Java Messaging Service) messages and RMI (Remote Method Invocation) calls. Other agents interact with each other indirectly by: (1) subscribing to the same input (i.e., conveyances, pallets, requests, supply centers, and routes); and, (2) manipulating the same output (i.e., the plan in the form of transfers and agent alerts). The interaction of the agents is depicted in Figure 6.3. Such a multi-agent architecture allows any number of agents to be developed individually and then integrated together as a whole with minimum effort. This leads to significant advantages for debugging operations and the extension of any individual agent.

Figure 6.3: Interaction of the agents

6.4 The Planning Agent
Built around the Tabu Search methodology, the Planning Agent is the backbone of the TRANSWAY® system’s planning capabilities and is responsible for performing planning and replanning operations under both static and dynamic conditions. A description of the Planning Agent’s system architecture, the plan creation process, the heuristic and exhaustive search strategies, and dynamic planning, follows below (a more detailed technical description is provided in Appendix A).

Figure 6.4: System architecture of the Planning Agent as a set of interactive services
6.4.1 System Architecture

The system architecture of the Planning Agent can be best described as a group of interactive services. In other words, the Planning Agent was designed and implemented as a group of modular components. Each component provides a specific type of service and its operations are independent from the operations of the other components. The following modular services have been implemented as the components of the Planning Agent (Figure 6.4):

- **Routing Service:** This service listens to changes within the graph-like structure of nodes and route segments. For each type of conveyance, an all-pairs shortest path matrix is computed and maintained using the Floyd-Warshall\(^{14}\) algorithm. Accordingly, agents are able to ask the Routing Service whether two locations are reachable using a specific conveyance type and, if yes: What is the shortest route? Agents may also ask the Routing Service to find alternate routes if some routes are impeded due to severe weather conditions or other threats (e.g., enemy attack).

- **Conveyance Service:** This service is used by the Mission Generator for determining which available conveyance is well suited for a given mission. Once a conveyance has been chosen, the Conveyance Service also selects an appropriate layout for the conveyance based on the availability layouts and the demands of the mission. Furthermore, the Conveyance Service also determines specifically what pallets or personnel will be loaded onto each conveyance.

- **Mission Generator:** This service creates feasible mission possibilities through the assistance of the Routing Service and the Conveyance Service. The Mission Generator first determines what requests need to be fulfilled and then asks the Routing Service for best routes for delivering the requested cargo or personnel. Once some feasible routes have been determined, the Mission Generator utilizes the Conveyance Service to locate a conveyance that is appropriate for the mission.

- **Mission Scheduler:** This service is responsible for scheduling missions without violating given scheduling constraints. The Mission Scheduler considers the current solution and then tries to fit new missions into the current solution’s schedule. It achieves this by repeatedly attempting to schedule new missions ensuring that they do not violate given scheduling constraints and then adding them into the solution set.

- **Search Service:** This service guides the Tabu Search process by invoking an appropriate search strategy. Based on the Tabu Search methodology two different search strategies have been developed and utilized in TRANSWAY® namely the Heuristic Search strategy and the Exhaustive Search strategy (see Section 6.4.3). The selected strategy will utilize the Mission Generator to generate missions and then find viable schedules for them using the Mission Scheduler. Finally, the Search Service converts those scheduled missions into *moves* that can be applied to the current solution.

The modularity of the Planning Agent exemplifies the advantage of an ontology-based object-oriented programming approach, which allows a complex system to be developed that is both maintainable and provides flexibility for future extensions.

\(^{14}\) The Floyd-Warshall algorithm also known as the Roy-Floyd algorithm, is a graph analysis algorithm for finding the shortest paths in a weighted, directed graph. It is an example of dynamic programming.
6.4.2 Plan Creation Process

The Planning Agent operates with the concept of a *move*. A *move* is essentially the effect of replacing a mission or missions with another mission or missions within the current plan (i.e., the current solution). The Planning Agent first creates a set of possible *moves*, and then selects a feasible *move* based on the states of the current scenario (e.g., the current scenario might have already changed) and the current solution. During the *move* creation phase, the Mission Generator identifies possible cargo deliveries from its supply locations to the requesting location. These deliveries then are combined to create some possible missions. A mission is represented as a list of actions. These actions are conveyance specific, including refueling, cargo loading, cargo unloading, and route traversing. When applying a *move* to a plan, the Planning Agent ensures that the mission being added or modified does not break any of the applicable business rule constraints. The business rule constraints used by the Planning Agent include:

- A conveyance travels at the rate of its cruising speed.
- A conveyance cannot exceed its pallet capacity.
- A conveyance cannot carry more than its weight limitations will allow.
- A conveyance will attempt to avoid impediments unless the path is only partially blocked.
- A conveyance cannot be in two places at the same time.
- A conveyance returns to its home base after its mission is completed.
- A conveyance will not travel when it is scheduled as unavailable.
- Aircraft at an airport will not exceed the working MOG\(^{15}\) limitation.
- Requested cargo will be delivered by the requested ETD (Earliest Time of Delivery) and not after the requested LTD (Latest Time of Delivery).
- Loading and unloading times are added to each mission according to the conveyance configuration.
- Transshipments require conveyances with different traversal types.
- No missions will be scheduled for departure before the earliest plan commencement time.

After the application of each *move* an evaluation function is used to determine whether the plan has been improved. The evaluation function is composed of a number of hierarchical factors (i.e., objective values) that represent the weight (i.e., the relative importance) of the planning criteria. In the current version of TRANSWAY\(^{®}\) the most important planning criterion is to minimize any supply shortfall. This applies to both requested cargo and requested personnel. The other criteria include minimizing the overall travel distance and estimated monetary cost of the plan. If the plan is improved then it is reported to the system user as the best-plan-so-far, while the Planning Agent continues searching for better plans.

---

\(^{15}\) MOG (Maximum on Ground) refers to the maximum number of aircraft that can be parked at an air facility at any one time. This is governed by both the size of the facility and the management of the available parking space (e.g., reservation of space for missions and other uses).
6.4.3 Heuristic and Exhaustive Search Strategies

As mentioned previously the Search Service of the Planning Agent utilizes two different search strategies: heuristic search; and, exhaustive search. When the heuristic search strategy is invoked, based on some heuristic metrics, the Planning Agent generates and selects moves that appear to be good candidates for a solution. This strategy allows the Planning Agent to rapidly explore a solution space and come up with an initial solution. However, the resultant solution will not necessarily be the best solution, because heuristic methods prune large portions of the search space. When the exhaustive search strategy is invoked, the Planning Agent performs a more thorough search by examining a large number of candidate moves. Any move that meets necessary constraints can be applied to the solution. In this way, the search can approach an optimal solution. However, the execution of the exhaustive search strategy can be unacceptably time consuming due to the \textit{NP-hard} nature of the problem.

In order to take advantage of the advantageous characteristics of both the heuristic and exhaustive strategies, the Planning Agent adopts a hybrid strategy that integrates the two strategies through a control mechanism. This mechanism allows the Planning Agent to monitor its own progress and to automatically choose an appropriate strategy for execution based on: (1) how often the search finds improved solutions; and, (2) how often the search is in a stagnant state. As a result, the Planning Agent is able to find some feasible solutions in a short amount of time and improved solutions if it is allowed to continue running. Figure 6.5 depicts the search flow of the hybrid strategy.

![Figure 6.5: Search flow of the hybrid heuristic and exhaustive search strategy](image)

6.4.4 Dynamic Planning

The Planning Agent is able to adapt to changes in a scenario while continuously planning. When a scenario is modified, the Planning Agent performs several steps to adapt to the changed conditions. The first step is to update the local representation of the scenario to reflect the change. Upon notification of a change, the Search Service identifies what data are no longer valid and adjusts them accordingly. In the second step the Planning Agent identifies which part of the solution is affected and removes invalid missions. For example, if a new impediment has been placed over a route that is being used by a mission, then any dependent mission becomes invalid and is removed from the solution. After that, the search continues its iterative process and repairs portions of the solution by attempting to replace them with new missions (Figure 6.6).
6.5 The Feasibility Agent

The Feasibility Agent provides feedback to TRANSWAY® users in cases where the Planning Agent does not have sufficient information to produce a legitimate plan. Feedback includes an explanation to the user of why some requirements are unable to be satisfied and suggestions for resolution. This is a typical example of human-computer collaboration. Through the assistance of the Feasibility Agent, a user is able to quickly adjust input data to allow the system to proceed smoothly.

The Feasibility Agent analyzes each requirement in a scenario and posts alerts to the user for the following problems:

- **No requirements:** There are no requirements in the scenario.
- **No sub-requirements:** A requirement has no sub-requirements.
- **Not enough supplies:** There are not enough items in the scenario to meet a cargo requirement.
- **No deliveries possible:** No conveyances can reach the requirement location from a supply center containing the requested items.
- **Cannot schedule:** No conveyances can delivery to the requirement location within the ETD and LTD constraints.

In the first version of TRANSWAY® the feasibility functionality was implemented within the Planning Agent. However, the increasing complexity of the Planning Agent required this functionality to be externalized. The creation of a new agent to handle the feasibility checks has resulted in faster user feedback and allows the Planning Agent to focus on planning and replanning capabilities.
The Feasibility Agent consists of three components: a Data Translator, a Constraint Checker, and an Alert Injector. The Data Translator monitors changes in a scenario. The Constraint Checker verifies the feasibility of each requirement and the Alert Injector posts new alerts to the TRANSWAY® user, removing old alerts that are no longer valid in the process (Figure 6.7).

6.6 Truth Maintenance Agent

The Truth Maintenance Agent verifies the consistency of a plan in the Domain Model and hence also in the database. In a multi-agent system, maintaining logical consistency of a database is a challenging task. Although each agent may work on a different aspect of the problem, all agents accomplish their tasks through subscribing to and manipulating the same database. Because the definition of logical consistency of the database may vary with respect to each agent, it is important to verify that the input of data from each agent into the database is consistent throughout the system as a whole.

The Truth Maintenance Agent relies on some specific constraints to check the consistency of a plan. In TRANSWAY® a plan is represented as a collection of transfer objects. When a constraint associated with a transfer object is not satisfied, the Truth Maintenance Agent marks the transfer object as invalid and posts an alert to the user. The Truth Maintenance Agent considers the following constraints and posts alerts when any of them is violated:

- **MOG violation**: The maximum on ground working constraint cannot be violated.
- **Traversal type violation**: A conveyance shall match its traversal type.
- **No type layout for conveyance**: A conveyance must have a type layout.
- **Unrequested cargo delivery**: Only requested cargo can be delivered.
- **Non-existent cargo loading**: The cargo loaded does not exist.
- **Invalid in-transit transfer end time**: The end time of an in-transit transfer has to be equal to the amount of time it takes the conveyance to traverse the route.
- **Impediment blocking traversal**: A transfer must not have an active impediment blocking its traversal.
- **A conveyance is double-booked**: A conveyance must not be double-booked for the same period of time.
- **A conveyance is limited by its range**: A conveyance cannot travel further than its maximum range.
- **Mission time limit exceeded**: A mission cannot exceed its maximum time limit.
- **Conveyance unavailability constraint violated**: A conveyance cannot be used when it is unavailable.
- **Passenger unavailability constraint violated**: A passenger cannot be transported when he or she is unavailable.

The Truth Maintenance Agent communicates with the database to keep track of the creation, modification, and deletion of objects. When approved transfers are affected by scenario changes, the consistency of each of the effected transfer object is checked. The Truth Maintenance Agent
performs tests that verify the validity of each aspect of the transfer object. If a test fails, then the transfer object is marked as invalid, and an alert is posted accordingly (Figure 6.8).

6.7 Impediment Agent

The Impediment Agent monitors route conditions in TRANSWAY® and notifies the Planning Agent if any route is impeded and/or the state of a previously impeded route has changed. Therefore, the Planning Agent is able to adjust its search to the updated route condition accordingly. An impediment is an obstruction applied to a route that can potentially hinder a conveyance traversing the route. The system currently supports two types of impediments, namely a weather impediment (e.g., storm) and a man-made impediment (e.g., military attack). Each impediment is represented as an area of impedance with a time window defining the duration of its influence.

The Impediment Agent listens to the creation, modification, and deletion of impediments in the Domain Model (and hence the database), and continuously checks for affected routes. If a route is found to be impeded by one or more impediments, a list of impediment time windows is calculated and maintained. A resultant time window contains impediment starting time and ending time, as well as a value representing how an impediment would affect a conveyance. Based on this knowledge, the Impediment Agent provides the Impediment Service to the Planning Agent in answering questions such as: Is a particular route impeded? If yes, when and how? Therefore, the Planning Agent is able to either schedule missions to avoid traveling on a route during the time it is impeded, or find some alternate routes that are free of impediments (Figure 6.9).
6.8 Related Work in the Military Domain

The military distribution problem is often represented as a constrained instance of the vehicle routing problem. Techniques for solving constrained optimization problems are discussed in (Barták 1999). Several classical heuristics for the vehicle routing problem are compared to metaheuristics-based techniques in (Laporte et al. 2000), and the results show that Tabu Search is the most successful metaheuristic approach. The Tabu Search techniques presented are able to quickly find optimum or near-optimum solutions for problem sets containing over 100 cities. However, some of the algorithms require user-controlled parameters for different problem sets to find good solutions.

A common optimization technique for the vehicle routing problem is metaheuristic search (Bräysy and Gendreau 2001, Laporte et al. 2000, Nanry and Barnes 2000, Rego and Roucairol 1995, Yang et al. 2004). The class of metaheuristics includes genetic algorithms, Tabu Search, and simulated annealing. A metaheuristic is an iterative algorithm utilizing domain-specific knowledge in the form of heuristics that are controlled by an upper level strategy (Blum and Roli 2003). Metaheuristics attempt to efficiently explore the search space while avoiding entrapment in local optima. In this way they are capable of discovering near-optimum solutions to problems that are intractable for exhaustive search techniques. Metaheuristics must make a trade-off between intensification and diversification of the search space. A search that only uses intensification techniques will converge to a solution quickly, but will not outperform a greedy heuristic. A search that only uses diversification techniques will explore a larger portion of the search space, but may fail to converge to a useful solution. Therefore, metaheuristics often utilize memory or induce randomness to guide the search process.

Group theory offers an efficient strategy for searching through move neighborhoods of combinatorial optimization problems (Barnes et al. 2004, Colletti and Barnes 2004, Crino et al. 2002). It provides a mathematical framework for characterizing metaheuristic search neighborhoods. A symmetric group of n letters (Sn) can be used to build and solve equations whose coefficients and variables are permutations (Barnes et al. 2004). These equations can be used to represent metaheuristic search neighborhoods for combinatorial optimization problems. Group theory has been combined with Tabu Search to solve the theater distribution vehicle routing and scheduling problem (Crino et al. 2002) and the aerial fleet refueling problem (Colletti and Barnes 2004). In these approaches group theory is used to construct abstract groups and move neighborhoods. An abstract group is a set of objects, along with a method of combining its elements that is subject to a few simple rules. Several abstract groups are combined to produce a feasible solution. Move neighborhoods allow the generation of abstract groups to prevent re-cycling, as well as entrapment in local optima. Group theory also reduces the number of user-controlled parameters required by the search.

6.8.1 Variations of the Vehicle Routing Problem

Relevant variations of the vehicle routing problem include the vehicle routing problem with time windows (Bräysy and Gendreau 2001), the pick-up and delivery problem (Nanry and Barnes 2000), the multi-terminal truck dispatch problem (Rego and Roucairol 1995), and the real-time vehicle routing problem (Yang et al. 2004). The vehicle routing problem with time windows expands the vehicle routing problem by adding restrictions to the delivery times for customers. In problem instances with hard time windows, the vehicles must visit locations during the
specified time windows. In problem instances with soft time windows, a penalty is given to vehicles that visit the locations outside of the time windows. A Tabu Search technique is presented by Bräysy and Gendreau (2001) that assigns a large penalty to solutions that violate the time window constraints. The penalty reinforces the search to discover feasible solutions. An initial solution is constructed using well-known heuristics and then improved with Tabu Search. The approach also uses random choices to diversify the search process. The results presented by Bräysy and Gendreau (2001) demonstrate that Tabu Search is well suited for the vehicle routing problem with time windows.

A reactive Tabu Search for the pick-up and delivery problem with time windows is presented by Nanry and Barnes (2000). The pick-up and delivery problem consist of assigning a fleet of homogeneous vehicles, starting from one initial hub, to a set of deliveries. Each delivery consists of picking up material at a predetermined location and delivering the material to its paired destination during the specified time window. This problem differs from the military distribution problem, because transshipments are not allowed. Transshipment is the delivery of goods to an intermediate location and from the intermediate location to another destination. Another difference is that the destination for the cargo is predetermined, which is not always true for military scenarios. A hierarchical search methodology is used to dynamically alter the neighborhoods and adjust the search trajectories.

Another variant of the vehicle routing problem is the multi-terminal truck dispatch problem (Rego and Roucairol 1995). This problem is similar to the pick-up and delivery problem with time windows, but considers social aspects as well as uncertain customer demand. Social constraints include the number of driving hours, rest periods, and speed limits. An initial solution is constructed using a classical heuristic and then improved using Tabu Search. The technique allows infeasible solutions, but a penalty is assigned to solutions that violate constraints. Therefore, it is possible for the search to cross through infeasible regions of the search space.

A real-time instance of the vehicle routing problem is considered by Yang et al. (2004). In the real-time vehicle routing problem, customer requests arrive continuously over time. The problem is decomposed into two stages. The first stage considers the off-line version of the search using a mixed integer programming formulation. The second stage considers a rolling horizon using repeated optimization of various instances of the off-line problem. The search improves when possible future jobs are taken into consideration. There is a trade-off between a more efficient solution that contains tight scheduling and a less optimal solution that absorbs small deviations (Barták 1999).

### 6.8.2 Simulation Versus Optimization Techniques

Current models addressing strategic mobility use aggregate network flow models, one-pass greedy approaches, and simple bounding techniques (McKinzie and Barnes 2004). The aspect that is lacking in current models is the use of optimization techniques. A simulation tool currently in use is the Mobility Analysis Support System (MASS) (Wu et al. 2003). MASS is a rule-based simulation model for scheduling military airlift. The MASS model selects the first available requirement and checks the first available aircraft to see whether the aircraft can deliver the requirement. The model then determines if the delivery is feasible by checking the capacity of the aircraft and verifying that the destination airbase can accommodate the aircraft. If the delivery violates a rule, then the simulation will attempt to use the next available aircraft to make the delivery. The process repeats until the first requirement is satisfied. The simulation then
attempts to satisfy the remaining requirements using the same process. The policy used by MASS is rule-based, which does not require a cost function for decisions. The model often produces inefficient solutions, but is deterministic and does not violate constraints contained in the rule set.

An optimization approach for solving the military airlift problem is the NRMO model (Baker et al. 2002). The NRMO model uses linear programming to solve a complex objective function based on several costs. An example cost is the range-payload curve, which specifies the weight that an aircraft can carry for a given distance. The model considers hard operating costs such as fuel and maintenance, as well as soft costs that encourage desirable behavior. The cost function penalizes deliveries of cargo that arrive after the delivery date as well as undelivered cargo. The model also offers a reward for aircraft that remain at an APOE, since these aircraft may be utilized for unforeseen contingencies (Wu et al. 2003).

The NRMO model consists of sets and subsets, such as sets of time periods, requirements, cargo types, aircraft types, and routes. The algorithm makes decisions by selecting a single element from several sets and forming an index combination. The index combination consists of an aircraft delivering a requirement on a route at a certain time. To reduce the problem to a tractable computation, a screening process removes infeasible index combinations. NRMO utilizes a cost model eliminating the need for extensive tables of rules. This allows the optimization model to solve several problem sets without recalibration of the rules or parameters. However, the linear programming model does have limitations. It is difficult to construct a cost function that produces the desired behavior in all possible situations. The need to specify behavior through a cost function is the main challenge for optimization techniques.

Bridging simulation and optimization models are stochastic programming models. The stochastic sealift deployment model SSDM (Morton et al. 2003) is a mixed-integer program designed to prepare for potential enemy attacks with probabilistic knowledge of the time, location, and severity of the attack. It uses a cost function rather than a set of rules, and therefore faces the same limitations of other optimization approaches such as NRMO. The SSDM model builds a scenario tree for different time periods when an attack may occur. The model allows a degree of uncertainty, but certain scenarios may result in an exponential run-time.

The optimizing simulator is a technique for achieving the flexibility of a simulation as well as the intelligence of an optimization (Wu et al. 2003). The optimizing simulator uses sets of rules the same way as traditional simulation techniques, but it is also iterative. The algorithm controls the amount of information available to the decision function, providing a natural bridge between optimization and simulation (Wu et al. 2003). One of the main goals of the technique is to incorporate a degree of uncertainty into the optimization process. This hybrid approach offers two principal advantages: it utilizes the optimization model’s ability to produce intelligent solutions to new problems sets; and, at the same time the simulation model allows the optimizing simulator to tolerate limited uncertainty.

**6.8.3 Transshipment Problems**

The problem tackled by TRANSWAY® includes transshipments, requiring the transfer of material from one vehicle to another at an intermediate location. The combination of the transshipment problem and the vehicle routing problem describes the real-world transportation environment more precisely. However, these problems are rarely examined as mathematical optimization
problems (Mues and Pickl 2005). Previous techniques have relied on flow networks to simulate transshipment problems (McKinzie and Barnes 2004). Flow networks are used to analyze the cargo moving through an individual location in a scenario. However, flow networks may result in bottlenecks. This problem occurs when a theater receives all cargo through a single hub, but the amount of cargo requested is greater than the hub throughput capacity. Once a bottleneck is identified, it may be possible to modify the flow network to avoid the problem. Therefore, flow network tools are more effective for analysis than optimization.

A mathematical model for the vehicle routing problem with time windows and transshipments is presented by Mues and Pickl (2005). Mathematical models allow for additional constraints that cannot be applied to flow networks, such as time windows. The model proposed by Mues and Pickl is generalized to represent the pick-up and delivery problem with time windows and is solved using mixed-integer programming. The problem is initially solved without the use of transshipments. Once the initial problem has been solved, a column generation technique is utilized to determine possible transshipment locations. The model produces high quality solutions, but the mixed integer programming technique becomes computationally expensive for large problem sets. This approach cannot be directly applied to the military supply problem, because the model is based on the pick-up and delivery problem. In military scenarios, the destination of material is often initially unknown.
7. The Thick Client User-Interface

The TRANSWAY® Thick Client is a Java based client that allows the user full interaction (as opposed to minimal interaction allowed by the Thin Client) with various objects, maps, reports, graphs, and alerts pertaining to the problem domain.

7.1 Overall Architecture

The Thick Client utilizes multiple ICDM components and services. Figure 7.1 shows the overall TRANSWAY® architecture with an emphasis on the Thick Client. Components and services that...
are not pertinent to the Thick Client have been grayed out and extra arrows have been added to show the specific dependencies and component interaction pertaining to the Thick Client.

The Thick Client is based entirely on ICDM’s Generic Space Generator (GSG) Framework, which is discussed in Section 7.2. The Thick Client also depends on GSG Mapping, discussed in Section 7.3, to display detailed images of the earth. GSG Mapping utilizes the Imagery Service to retrieve the required map images. In order to interact efficiently with the Domain Model, the Thick Client utilizes a set of Facades, the Subscription Service, and the Persistence/Query Service (see Chapter 5).

The Facades offer a higher-level interface to components in the Thick Client than do objects returned by the Persistence/Query Service. The Subscription Service allows components within the Thick Client to register for creations, deletions, and modifications to the Domain Model. For example, when a user creates a Supply Center on one client, any other client that is registered for these creations will receive a notification. For performance purposes, the Thick Client has limited the number of subscriptions it holds. For instance, reports and charts do not stay synchronized with the Domain Model. The Persistence/Query Service allows components within the Thick Client to persist and query for objects without the use of the Facades. This programming alternative proved useful under certain circumstances during the development of TRANSWAY®.

As shown in Figure 7.2, the Thick Client is divided into four domains. The Map Domain is the parent domain and allows users to create, delete, modify, and interact with objects in the system. The Transfer Model Domain allows users to view detailed information about current plans. The Charts Domain provides users with charts pertaining to supplies, conveyances, plans, and so on. Finally, the Agents Domain allows users to view the status and explanations provided by the agents.

![Figure 7.2: Domains within the Thick Client](image)

### 7.2 Generic Space Generator (GSG) Framework

ICDM’s Generic Space Generator (GSG) is designed as a client architecture to facilitate the rapid development of Graphical User Interfaces (GUI) through the use of the Model-View-Controller (MVC) pattern. The MVC pattern allows for the separation of the user-interface from the data model and can be specified from a combination of generic GSG GUI components. The main GSG
GUI component is the Viewer, which allows for the fast graphical display and manipulation of an application’s data model. The GSG Viewer was built on the Java Binding for the OpenGL API (JOGL), shown in Figure 7.3, and allows for the optimized performance of both two-dimensional and three-dimensional graphics (Wright and Sweet 2000).

Figure 7.3: The GSG-JOGL Binding interface

The more recent releases of Java available at the time this version of TRANSWAY® was developed included the ability to pass arrays to C++ in the form of ByteBuffer. These contain a pointer to the represented array, as well as information such as length. The ability to pass arrays efficiently is important since many graphical functions require large arrays of vertex and line information to be passed to the video card.

The principal components of the GSG architecture and their relationships are depicted in the system diagram shown in Figure 7.4 and briefly explained below.

- **Domain XML**: Describes a set of entities and reusable components, as well as the layout and menus of components. Passed to the Domain Loader in order to create a Domain component.
- **Domain Loader**: Transforms the Domain XML into a Domain object. Initializes any Viewer(s) and the Widget Manager.
- **Domain**: Describes a set of entities and reusable components. Captures a log of changes that allows for rollback capabilities.
- **Domain Model**: Contains all graphical and logical entities. Components, such as the Viewer, listen for changes to the Domain Model in order to determine the existence of entities instances.
- **Entities**: Objects that adhere to the Java Bean programming model. Provide functions to get and set properties and allow notification of property changes through PropertyChangeEvent.
- **Graphical Entities**: Entities that know how to draw themselves in the Viewer. Properties of these entities can be viewed and edited.
- **Logical Entities**: Entities that are purely informational and cannot be displayed in the Viewer. Properties of these entities can be viewed and edited.
- **Local Agents**: Provides data validation and notify users of potential problems with entities within the domain.
- **Controllers**: Responds to property changes of entities, performing behind the scenes services.
- **GSGActions**: Reusable units of functionality that perform a service in reaction to certain events such as user interactions.
- **Widgets**: Reusable GUI components that are responsible for the primary GSG display capabilities.
- **Widget Manager**: Main window or frame that is always visible. The Widget Manager is responsible for the layout and docking of inner windows. Every GSG application has one Widget Manager.
- **Viewer**: Displays graphical representation of the entities that exist in the Domain Model and allow the user to manipulate entities using the Interactions.
- **Interactions**: Handles user mouse actions for the Viewer in situations such as movement, placement and modification of entities.
- **Domain Views**: Provides different types of visual representations of the entities in the Domain Model (e.g., a hierarchical tree layout). Entities can usually be modified or edited from this view.

![Diagram of GSG Architecture](image)

**Figure 7.4**: Overview of the GSG architecture

### 7.3 GSG Mapping

GSG Mapping is an ICDM component used to display maps while accessing the Imagery Service. This component also provides utility tools to query for information about maps, ranging from simple data such as distance and elevation to more complex data such as transportation routes.
and coastal highlighting. Through the extension of abstract map handlers and download requests GSG Mapping is able to retrieve maps from any location (i.e., from disk or the Imagery Service).

The maps themselves are implemented using a pyramid tile-based format, as discussed in previously in Chapter 5. Each map is essentially a collection of 256 x 256 pixel tiles. As the viewer zooms closer to a map, the level of detail is enhanced by loading more detailed tiles. Each level in the pyramid consists of a different level of tiling detail. The topmost level of the pyramid consists of tiles with the least resolution, and the bottommost consists of tiles with the highest resolution. Each tile on any one level can then be represented by four more tiles with four times the resolution in the subsequent level below it. As more tiles are loaded onto the Viewer, each tile is cached locally for fast rendering.

GSG Mapping also provides several general tools, such as a cumulative distance tool and a coordinate conversion tool. For more complicated data, GSG Mapping contains a tool that allows users to use vector layers to query metadata about a selected area. This information is retrieved directly from the Imagery Service and can include data about transportation routes, power transmission lines, dams, rivers, elevation, bridges, cities, and so on.

The GSG Mapping component can also display maps using different projections. This allows the user to switch between projections. Some of the currently supported map projections are Mercator, Bonne, Cassini, Cylindrical Equal-Areas, Gnomonic, and Robinson.

7.4 Map Domain

The Map Domain is the Thick Client’s parent domain (see Figure 7.2) and allows users to create, delete, modify, and interact with objects in the system. Components in the Map Domain can be characterized as GSG-based components (i.e., derivatives of components discussed in Section 7.2). Since the architecture of the Map Domain is based directly on the GSG architecture, it is adequately represented by Figure 7.4 (see Section 7.2). The principal GSG-based components are briefly described below.

- **Domain XML:** One Domain XML file exists for the Map Domain. It describes all of the GSG-based components to be loaded by GSG’s Domain Loader.

- **Graphical Entities:**
  - Map Layers – All map layers are considered Graphical Entities. Example map layers are the Route Layer, the Text Layer, the Supply Layer, and the Vector Map Layer.
  - Graphical Facade – Any Facade that can be added to the map is considered a Graphical Entity. Current Graphical Facades are Supply Centers, Waypoints, Route Segments, Notes, Weather Impediments, Other Impediments, Areas of Interest, and Areas to Ignore.

- **Logical Entities:**
  - Logical Facades – Any Facade that is created but cannot be added to the map is considered a Logical Entity (e.g., Cargo, Organizations, Requirements).
• **GSGActions:**
  o Imports (File menu) – Allows the user to restore Graphical and Logical Facades that were previously exported. Each import accepts an XML file, verifies its format against the appropriate schema, and creates database objects based on file data.
  o Exports (File menu) – Allows the user to save Graphical and Logical Facades that can be imported at a later time. Each export queries the database for necessary data and writes an XML file using the appropriate schema.
  o Reports (Reports menu) – Allows the user to create, view, sort, filter, delete, and modify Graphical and Logical Facades.

• **Widgets:**
  o Agent Widget – Displays agent icons along with the number of unacknowledged alerts.
  o Bean Selection Widget – Displays icons of Graphical Facades that can be added to the Viewer, allowing the user to select the appropriate icon and immediately click on the Viewer for creation.
  o Bean Properties Widget – Displays properties of the currently selected Graphical Entity, allowing the user to edit properties if desired.
  o Layers Widget – Displays the list of layers that can be set to be visible along with their properties.

• **Viewer:** Contains Graphical Entities, allowing the user to zoom and pan the visible Map Layers and Graphical Facade Beans.

7.5 Transfer Model Domain

The Transfer Model Domain allows users to view detailed information about current plans in the system. This is accomplished through two child domains, namely: the Animation Domain; and, the Transfer Domain. The Animation Domain contains a Viewer showing the Waypoints, Supply Centers, and Route Segments on which the user can animate a plan. The Transfer Domain contains a Viewer showing a Gantt chart of plans in the system, allowing the user to zoom, pan, and obtain information about particular aspects of a plan.

Similar to the Map Domain, components in the Transfer Model Domain can also be characterized as GSG-based components. The principal components include:

• **Domain XML:** Three Domain XML files exist for the Transfer Model Domain. The first is the parent domain XML file that describes how the other domains should interact and be laid out. The second and third are the child domain XMLs (for the Animation and Transfer Domains). All of the files describe the GSG-based components to be loaded by GSG’s Domain Loader.

• **Widgets (Transfer Domain):** The Bean Tree Widget displays a logical tree of objects in the Transfer Domain Viewer.

• **Viewer (Animation Domain):** Contains the Waypoints, Supply Centers, and Route Segments on which the user can animate a plan.
• **Viewer (Transfer Domain):** Contains a Gantt chart of plans in the system, allowing the user to zoom, pan, and obtain information about a particular aspect of a plan.

### 7.6 Charts Domain

The Charts Domain allows users to view charts pertaining to supplies, conveyances, plans, and so on. Again, components in the Charts Domain can also be characterized as GSG-based components. The significant components in the Charts Domain are listed below and described briefly.

- **Domain XML:** One Domain XML file exists for the Charts Domain. It describes all of the GSG-based components to be loaded by the GSG Domain Loader.

- **Graphical Entities:**
  - Chart Graphics – All lines, bars, and text objects that are displayed in the Viewer are considered Graphical Entities.

- **Logical Entities:**
  - Chart Properties – An editable entity containing properties such as chart color and font. The properties are propagated to the Graphical Entities in the Viewer via `PropertyChangeEvent`s.

- **GSGActions:**
  - Supply Centers Summary (Charts menu) – Executes a chain of `GSGActions`, described as follows: get all Supply Centers; allow user to select Supply Centers of interest; display a modal status dialog; convert Supply Center input to summary; create bar chart; and, display bar chart in the Viewer.
  
  - Outgoing Weight at Supply Centers (Charts menu) – Executes a chain of `GSGActions`, described as follows: get all Supply Centers; allow user to select Supply Centers of interest; display a modal status dialog; convert Supply Center input to outgoing weight at Supply Centers; create bar chart; and, display bar chart in the Viewer.
  
  - Incoming Weight at Supply Centers (Charts menu) – Executes a chain of `GSGActions`, described as follows: get all Supply Centers; allow user to select Supply Centers of interest; display a modal status dialog; convert Supply Center input to incoming weight at Supply Centers; create bar chart; and, display bar chart in the Viewer.
  
  - Weight By Type (Charts menu) – Executes a chain of `GSGActions`, described as follows: get all Supply Centers; allow user to select one Supply Center of interest; display a modal status dialog; convert Supply Center input to weight by type at Supply Centers; create bar chart; and, display bar chart in the Viewer.
  
  - Weight By Commodity Code (Charts menu) – Executes a chain of `GSGActions`, described as follows: get all Supply Centers; allow user to select one Supply Center of interest; display a modal status dialog; convert Supply Center input to...
weight by commodity code at Supply Centers; create bar chart; and, display bar chart in the Viewer.

- Conveyance Usage By Plan (Charts menu) – Executes a chain of \textit{GSGActions}, described as follows: get all \textit{PlanIDs}; allow user to select \textit{PlanIDs} of interest; display a modal status dialog; convert \textit{PlanID} input to conveyance usage by plan; create line chart; and, display line chart in the Viewer.

- Conveyance Usage By Route (Charts menu) – Executes a chain of \textit{GSGActions}, described as follows: get all Routes; allow user to select Routes of interest; display a modal status dialog; convert Routes input to conveyance usage by route; create line chart; and, display line chart in the Viewer.

- Gantt Chart – Conveyance Usage By Route (Charts menu) – Executes a chain of \textit{GSGActions}, described as follows: get all Routes; allow user to select Routes of interest; display a modal status dialog; convert Routes input to conveyance usage by route Gantt chart; create Gantt chart; and, display Gantt chart in the Viewer.

**Viewer:** Contains chart Graphical Entities, allowing the user to zoom and pan the visible chart.

### 7.7 Agents Domain

The Agents Domain allows users to view agent output and status. Components in the Agents Domain can again be categorized as GSG-based components. The significant components in the Agents Domain are listed below and described briefly.

- **Domain XML:** One Domain XML file exists for the Agents Domain. It describes all of the GSG based components to be loaded by the GSG Domain Loader.

- **Widgets:**
  - Agent Status Panel Widget – Displays the current status of each agent (e.g., planning, idle, or initializing) and allows the user to ping each agent to force a connection if necessary.
  - Planning Agent Panel Widget – Displays a console to the user containing error and progress output by the Planning Agent. Also, displays a graph showing the Solution Objective Value.
  - Feasibility Agent Panel Widget – Displays a console to the user containing error and progress output by the Feasibility Agent.
  - Truth Maintenance Panel Widget – Displays a console to the user containing error and progress output by the Truth Maintenance Agent.
  - Impediment Agent Panel Widget – Displays a console to the user containing error and progress output by the Impediment Agent.
8. The TRANSWAY® Web Client

The Web Client or Thin Client allows the user to interact with a limited subset of the functionality provided by the Thick Client described in the previous chapter. Like the Thick Client, it utilizes multiple ICDM components and services to allow the user to interact with various objects, maps, and reports. Figure 8.1 depicts the TRANSWAY® overall architecture with an emphasis on the Thin Client. Components and services that are not pertinent to the Thin Client have been grayed out and extra arrows have been added to show Thin Client specific dependencies and component interactions.

![Overall TRANSWAY® architecture with emphasis on the Thin Client]

---

Figure 8.1: Overall TRANSWAY® architecture with emphasis on the Thin Client
The Thin Client utilizes the Imagery Service to retrieve map images, as well as using the Facades and the Persistence/Query Service to interact with the Domain Model. The Facades offer a higher-level interface to components in the Thin Client than do objects returned by the Persistence/Query Service. They allow Thin Client developers to utilize one code base for creating, updating, refreshing, and maintaining TRANSWAY® objects, thereby reducing the requirement for complex transactions and queries. The Persistence/Query Service allows components within the Thin Client to persist and query for objects as an alternative to the use of Facades. Developers have found this alternative useful under particular circumstances.

The Thin Client makes use of the Backbase JSF Edition framework. This framework combines an AJAX development framework and standard Java technology and allows developers to create web interfaces for Java applications. Utilizing this framework, as well as JavaScript, HTML, and CSS files, the Thin Client consists of three principal components, namely: Map Viewer; Requests; and, Reports.

**Map Viewer:** The Map Viewer component is an interactive JavaScript component that is embedded within the Backbase framework of the Thin Client. This component allows users to view imagery representing the earth at different locations and heights relative to the earth's surface. The Map Viewer can also display additional imagery that is overlaid on the map imagery to provide the user with a visual representation of objects such as supply centers and routes. In order to display the appropriate imagery, the Map Viewer component takes user input that is used to retrieve information from the Imagery Service. The information is then returned to the client and displayed to the user.

**Requests:** The Requests component allows the user to view and submit requests for resources. Backbase components on the page take input from the user and then utilize the Facades to interact with the Database and return necessary information to the user.

**Reports:** The Reports component allows the user to view the resources at a particular supply center in a bar graph. User input specifying what supply center to graph is used to return a custom JavaScript component that represents the data in a bar graph.

---

16 Backbase is a collection of widgets that facilitate the implementation of web interfaces. Server-side support is tightly integrated with Java platforms such as the Java-based web application framework JavaServer Faces (JSF) and Struts. JSF uses a component-based approach in which the state of components is saved when the client requests a new page and restored when the request is returned.
9. Server Deployment

The TRANSWAY® suite of tools is currently deployed with four separate servers, referred to as the Web Server, the Data Server, the Map Server, and the Agent Server. The deployment diagram shown in Figure 9.1 depicts a typical configuration consisting of the four servers, the software components executing on each server, representative client machines (i.e., Client Machine 1 and Client Machine 2), and the specific connections that exist between the servers and the clients.

Figure 9.1: TRANSWAY® deployment diagram

The Web Server allows clients executing a Web 2.0 client (e.g., Client Machine 1) to run the Thin Client web application via HTTP through Apache Tomcat17. Besides communicating with client machines, the Web Server communicates with the Map Server and the Data Server. The

---

17 Apache Tomcat is a web container or application server developed at the Apache Software Foundation that implements the servlet and the Java Server Pages specifications of Sun Microsystems.
Web Server communicates with the Map Server through HTTP in order to request and receive map images from the Imagery Service. Communication between the Web Server and the Data Server is accomplished through a JDBC\(^\text{18}\) connection in order to persist data in the TRANSWAY\(^\text{®}\) database and to request and receive query results from the TRANSWAY\(^\text{®}\) database.

The Data Server runs the TRANSWAY\(^\text{®}\) database via Oracle. It listens for and answers JDBC requests from the Web Server, the Agent Server, and clients running the Thick Client (e.g., Client Machine 2).

The Map Server allows clients to utilize the Imagery Service through Apache Tomcat. It listens for and answers HTTP requests from the Web Server and clients running the Thick Client (e.g., Client Machine 2).

The Agent Server runs the Scenario Server, all of the agents, the ActiveMQ message broker, and ApacheDS. The Agent Server communicates with the Data Server through a JDBC connection in order to persist data in the TRANSWAY\(^\text{®}\) database and to request and receive query results from the TRANSWAY\(^\text{®}\) database. The ActiveMQ message broker manages messaging between clients running the Thick Client (e.g., two instances of Client Machine 2) via the Java Message Service (JMS). While Figure 9.1 does not show this explicitly, it should be noted that ApacheDS communicates with ActiveMQ and the Scenario Server via LDAP\(^\text{19}\).

All of the TRANSWAY\(^\text{®}\) servers are deployed behind a firewall. Figure 9.2 depicts a variety of different client machines (e.g., a laptop, desktop, or other computer or server) communicating with the TRANSWAY\(^\text{®}\) servers through a firewall. The diagram also depicts the connections that exist between the servers (see also Figure 9.1).

Figure 9.2: Deployment of TRANSWAY\(^\text{®}\) in a networked environment

\(^{18}\) JDBC is an API for the Java programming language that provides methods for querying and updating data in a relational database.

\(^{19}\) The Lightweight Directory Access Protocol (LDAP) is an application protocol for querying and modifying directory services running over TCP/IP.
10. Security

For any service-oriented application, such as TRANSWAY®, which implements some functionality for clients across a network (e.g., Internet, NIPRNET or SIPRNET) there is the need to ensure that the output from the service will be received only by authorized clients and that the incoming requests have not been altered at any point between a client and the service. Clients, on the other hand, need to be assured that they are connected to an authentic server and that the responses that they receive have not been intercepted and modified as they are passed back from the server.

The ability to provide this level of security is more easily achieved in a secure network such as the SIPRNET, where not only the access environment (i.e., physical facility) but also the human operators who have access to the network through client applications are carefully screened, monitored, and controlled. In non-secure networks, such as the NIPRNET and the Internet, the risks may be greater because even a much lower level of security may be more difficult to achieve.

TRANSWAY® was designed to provide a reasonable level of information assurance with the goal of continually improving security as the system matures. Password authentication is supported for all clients and services accessible outside the firewall, as a means of establishing identity. Additionally, all connections to services accessible through the firewall enforce integrity checking and are encrypted to preserve privacy and to ensure that the transfers have not been altered. Additionally, current plans include provisions for PKI-functionality to provide stronger authentication and role-based authorization to control access to resources.

Security measures have been implemented throughout the TRANSWAY® architecture at various levels, including: network; Java client; web client; web services; server operating system; Oracle database; ActiveMQ; ApacheDS; and, Tomcat. These security measures are described in more detail below.

10.1 Network Security

All Internet connections are encrypted and integrity checking is enforced. Servers are behind a firewall, with the provision that only connections from known IP addresses are allowed and only required ports are open.

10.2 Java Client Security

Java client security is ensured through password authentication and the enforcement of password complexity, as well as the following provisions:

- JDBC connections are encrypted (i.e., 3DES 168-bit encryption) and integrity checking (i.e., MD5 message digest) is enforced via Oracle Advanced Security.
- JMS connections are established only via Secure Sockets Layer (SSL).
- JNDI connections are established only via SSL.
- Web services connections are established only via SSL.
10.3 Web Client Security
Web client security is ensured through password authentication and the enforcement of password complexity, as well as the following provisions:

- Connections are only established via SSL.
- No authentication information is stored in cookies.\(^{20}\)

10.4 Web Services Security
The following measures have been taken to ensure web services security:

- Token-based authentication is required.
- Password authentication is required to retrieve a token.
- Password complexity is enforced.
- Only SSL connections may be established.

10.5 Server Operating System Security
The following measures have been taken to ensure server operating system security:

- The number of operating system users is limited.
- The privileges of the operating system accounts on the host are limited to the fewest and least powerful privileges required for each user.
- All unnecessary operating services are disabled.

10.6 Oracle Database Security
Apart from password authentication, including the enforcement of password complexity, the locking of user accounts after five consecutive failed log-in attempts, the locking and expiration of default user accounts, and the changing of default user passwords, the following Oracle database security measures have been implemented:

- Data dictionary protection is enabled such that only privileged database users may access the data dictionary.
- Only necessary privileges are granted for database user accounts.
- Modifying the default permissions for the installation directory or its contents even by privileged operating system users is not permitted.
- Database paths or files may only be modified by privileged operating system users.
- The Oracle Connection Manager is deployed for accessing the database from outside the firewall. In addition only encrypted connections and only those from known IP addresses may be established.
- Oracle executes in user-mode under an unprivileged user.

---

\(^{20}\) Cookies are files stored on a user's computer that identify a Personal Computer (PC) to a Web server. They typically authenticate or identify a registered visitor of a Web site or are part of a login process.
• Current patches have been applied.

10.7 *ActiveMQ* Security

In addition to password authentication and the enforcement of password complexity, the following measures have been implemented to ensure *ActiveMQ* security:

• Only SSL connections may be established.
• *ActiveMQ* executes in user-mode under an unprivileged user.
• Current patches have been applied.

10.8 *ApacheDS* Security

In addition to password authentication and the enforcement of password complexity, the following measures have been implemented to ensure *ApacheDS* security:

• Only SSL connections may be established.
• *ApacheDS* executes in user-mode under an unprivileged user.
• Current patches have been applied.

10.9 *Tomcat* Security

The following measures have been taken to ensure Tomcat security:

• Only SSL connections may be established.
• Tomcat executes in user-mode under an unprivileged user.
• Current patches have been applied.
11. References and Bibliography


Apache ActiveMQ Website: http://activemq.apache.org.

ApacheDS Website: http://directory.apache.org.

Apache Tomcat Website: http://tomcat.apache.org.


Battiti R. and G. Tecchiolli (1994); ‘The Reactive Tabu Search’; ORSA Journal on Computing, 6(2) (pp. 126-140).


Chandrasekaran, B., J. Josephson and V. Benjamins (1999); ‘What are Ontologies, and Why Do We Need Them?’; IEEE Intelligent Systems, 14(1), January/February.


Crino J., J. Moore, J. Barnes and W. Nanry (2002); ‘Solving the Theater Distribution Vehicle Routing and Scheduling Problem Using Group Theoretic Tabu Search’; Air Force Institute of Technology, University of Texas at Austin.


Glover F. (1990); ‘Tabu Search: A Tutorial’; Interfaces, 20 (pp. 74-94).


Hoffman K. and M. Padberg (2005); ‘Combinatorial andInteger Optimization’; Technical Report, George Mason University, New York University.

IHMC (2006); ‘Cmap Tool, Version 4.03’; http://cmap.ihmc.us/download/


Nanry W. and J. Barnes (2000); ‘Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search’; Transportation Research, 34 (pp. 107-121).


Pohl J. (2006); ‘Knowledge Management Enterprise Services (KMES)’; CDM Technologies, Inc., San Luis Obispo, California.


Pohl K. (2001); ‘Perspective Filters as a Means for Interoperability Among Information-Centric Decision-Support Systems’; Office of Naval Research (ONR) Workshop hosted by the CAD Research Center in Quantico, VA, June 5-7.


Schrady D. (1999); ‘Combatant Logistics Command and Control for the Joint Force Commander’; Navy report.


Talbi E. G. (2002); ‘A Taxonomy of Hybrid Metaheuristics’; Journal of Heuristics, 8 (5), September (pp.541-564).


### 12. Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DES</td>
<td>Triple Data Encryption Standard</td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AOA</td>
<td>Add-on-Armor</td>
</tr>
<tr>
<td>AOR</td>
<td>Area of Responsibility</td>
</tr>
<tr>
<td>APOD</td>
<td>Aerial Port of Debarkation</td>
</tr>
<tr>
<td>APOE</td>
<td>Aerial Port of Embarkation</td>
</tr>
<tr>
<td>BEL</td>
<td>Business Engine Layer</td>
</tr>
<tr>
<td>BRL</td>
<td>Business Rule Layer</td>
</tr>
<tr>
<td>BMNG</td>
<td>Blue Marble Next Generation</td>
</tr>
<tr>
<td>CADRC</td>
<td>Collaborative Agent Design Research Center</td>
</tr>
<tr>
<td>CADRG</td>
<td>Compressed ARC Digitized Raster Graphics</td>
</tr>
<tr>
<td>CDE</td>
<td>Corporate Data Environment</td>
</tr>
<tr>
<td>CIB</td>
<td>Controlled Image Base</td>
</tr>
<tr>
<td>CICE</td>
<td>Corporate Information-Centric Environment</td>
</tr>
<tr>
<td>COA</td>
<td>Course of Action</td>
</tr>
<tr>
<td>COI</td>
<td>Community of Interest</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>DAO</td>
<td>Data Accessor Object</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>DDL</td>
<td>Data Description Language</td>
</tr>
<tr>
<td>DDOOC</td>
<td>Deployment and Distribution Operations Center</td>
</tr>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>DODAAC</td>
<td>Department of Defense Activity Address Code</td>
</tr>
<tr>
<td>ETD</td>
<td>Earliest Time of Delivery</td>
</tr>
<tr>
<td>GDAL</td>
<td>Geospatial Data Abstraction Library</td>
</tr>
<tr>
<td>GEOLOC</td>
<td>Geographic Location</td>
</tr>
<tr>
<td>GEOTIFF</td>
<td>Geostationary Earth Orbit Tagged Image File Format</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GSG</td>
<td>Generic Space Generator</td>
</tr>
</tbody>
</table>
GTN  Global Transportation Network
GUI  Graphical User Interface
GUIL  Graphic User Interface Layer
HQL  Hibernate Query Language
HTML  Hypertext Markup Language
HTTP  Hyper-Text Transfer Protocol
HTTPS  Hypertext Transfer Protocol over Secure Sockets
IATA  International Air Transport Association
ICAO  International Civil Aviation Organization
ICDM  Integrated Cooperative Decision Making
ICODES  Integrated Computerized Deployment System
ID  Identification
IHMC  Institute for Human and Machine Cognition
IP  Internet Protocol
JDBC  Java Database Connectivity
JDDOC  Joint Deployment and Distribution Operations Center
JFC  Joint Forces Commander
JFCT®  Joint Forces Collaborative Toolkit
JMS  Java Messaging Service
JNDI  Java Naming and Directory Interface
JOA  Joint Operating Area
JOGL  Java Open Graphics Library
JOPES  Joint Operation Planning and Execution System
JPEG  Joint Photographic Experts Group
JSF  Java Server Faces
KMES®  Knowledge Management Enterprise Services
LDAP  Lightweight Directory Access Protocol
LP&R  Logistics Planning and Re-planning Services
LTD  Latest Time of Delivery
MD5  Message Digest 5
MOG  Maximum on Ground
MOGp  Maximum on Ground - Parking
MOGw  Maximum on Ground – Working
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NIPRNET</td>
<td>Non-Secure Internet Protocol Router Network</td>
</tr>
<tr>
<td>NITF</td>
<td>National Imagery Transmission Format</td>
</tr>
<tr>
<td>NP</td>
<td>Non-Deterministic Polynomial</td>
</tr>
<tr>
<td>NSN</td>
<td>National Stock Number</td>
</tr>
<tr>
<td>OAL</td>
<td>Object Access Layer</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Data</td>
</tr>
<tr>
<td>OML</td>
<td>Object Management Layer</td>
</tr>
<tr>
<td>OTL</td>
<td>Object Transport Layer</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Open Graphics Library</td>
</tr>
<tr>
<td>OpenTS</td>
<td>Open Tabu Search</td>
</tr>
<tr>
<td>ORM</td>
<td>Object Relational Mapping</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>POD</td>
<td>Port of Debarkation</td>
</tr>
<tr>
<td>POE</td>
<td>Port of Embarkation</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>SIPRNET</td>
<td>Secret Internet Protocol Router Network</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Service-Oriented Architecture Protocol</td>
</tr>
<tr>
<td>SPOD</td>
<td>Surface Port of Debarkation</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSA</td>
<td>Supply Support Activity</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>SSDM</td>
<td>Stochastic Sealift Deployment Model</td>
</tr>
<tr>
<td>SSN</td>
<td>Social Security Number</td>
</tr>
<tr>
<td>TCN</td>
<td>Transportation Control Number</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol / Internet Protocol</td>
</tr>
<tr>
<td>TIRAC®</td>
<td>Toolkit for Information Representation and Agent Collaboration</td>
</tr>
<tr>
<td>TL</td>
<td>Transport Layer</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery and Integration</td>
</tr>
<tr>
<td>UIC</td>
<td>Unit Identification Code</td>
</tr>
<tr>
<td>ULN</td>
<td>Unit Line Number</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>US</td>
<td>United States of America.</td>
</tr>
<tr>
<td>USTRANSCOM</td>
<td>United States Transportation Command</td>
</tr>
<tr>
<td>VPF</td>
<td>Vector Product Format</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>VPF</td>
<td>Vector Product Format</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
</tbody>
</table>
Appendix A: Design and Implementation of the Planning Agent

The purpose of this Section is to provide a more detailed description of the design, implementation, and performance of the Tabu-based Planning Agent in TRANSWAY®. The knowledge modeling toolkit, CmapTools (IHMC, 2006), has been used for documenting and generating some of the information presented in this Appendix.

A-1 Problem Definition

The application domain of the Planning Agent in TRANSWAY® is a dynamic vehicle scheduling and routing problem that can be described as follows:

A topographic map contains nodes and routes. Nodes associate with some supplies and/or conveyances. Routing conditions may change over time. Given a set of requirements for supplies to be delivered to some nodes within certain time windows, find a plan for conveyances that satisfies the requirements. Routing conditions and requirements may change dynamically over time.

Such a problem essentially consists of two separate but inter-related problem states, namely: an initial planning state; and, a re-planning state. The planning state can be viewed as the solution of either a Multi-Processor Scheduling problem or a Combinatorial Optimization problem\(^{21}\), depending on how this state is represented. Although the input to the planning state problem can be viewed as static, it is clear that finding an optimal solution is impractical in reality due to its NP-Completeness. A re-planning state occurs when the input to the problem, such as a requirement or a route condition, changes after a solution has been found (or partially found). Such a change forces either an existing solution to be modified, or a completely new solution that satisfies a new set of constraints to be created.

Depending on the specific instance of the planning state problem there may not exist a solution that can satisfy all of the requirements while satisfying all of the constraints. At other times there may exist multiple solutions, with some of these solutions being better than others based on a given set of assessment criteria. In the case of the re-planning state, a solution may sometimes be found by modifying a previous solution, while more commonly it will be necessary to create a new solution. This presented the following specific challenges to the developers of the Planning Agent:

1. **Planning state:** How to determine if a solution to the problem exists (i.e., there may be no possible solution)?
2. **Planning state:** Having determined that a solution is possible, how to find a feasible solution within a reasonable time frame (since searching for an optimal solution is clearly out of the question)?
3. **Re-planning state:** When to modify an existing solution to obtain a new solution, and when to create a new solution?

\(^{21}\) Both problems are NP-Complete and all instances of NP-Complete problems can be represented identically in one form or another.
4. **Re-planning state:** How to generate a new solution efficiently by modifying an existing solution?

The next section describes how these challenges are addressed by the Planning Agent using Tabu Search Framework.

**A-2 Tabu Search Framework**

The Planning Agent currently utilizes Tabu Search Framework to solve the dynamic vehicle scheduling and routing problem. Tabu Search is a mathematical optimization method. It uses a local or neighborhood search procedure to iteratively move from a solution \( x \) to a solution \( x' \) in the space \( N(x) \), the neighborhood of \( x \), until some termination criterion has been satisfied. In order to escape local optimality, Tabu Search modifies the neighborhood structure of each solution as the search progresses. The solution’s admitted \( N^*(x) \), the new neighborhood, is determined through the use of special memory structures (i.e., Tabu List). The search then proceeds by iteratively moving from a solution \( x \) to a solution \( x' \) in \( N^*(x) \) (Figure A-1).

![Solution Space](image)

**Figure A-1: The underlying concept of the Tabu Search method**

**Figure A-2: Iteration in the Tabu Search framework (Harder 2001)**

Some important features of the Tabu Search method include:

1. It applies an incremental optimization strategy. A search can start with any possible solution of a problem or even an empty solution and then attempt to improve the solution iteratively (Figure A-2). The quality of a solution relies heavily on how it is evaluated against a former solution. Each incremental solution is a candidate solution.

2. It utilizes a Tabu List memory structure. Tabu List prevents a search process from being trapped in a local space where the search has already visited. Therefore the search is able to continuously explore new potential solution neighborhoods for improvement.

3. It looks for a *good enough* solution instead of the optimum solution. Nevertheless, if given enough time, it can theoretically find the optimum solution, if such a solution exists. Using certain Tabu List strategies it is possible to systematically explore the solution space to obtain a more guaranteed convergence (Glover 2001).
4. It has the ability to adapt to changes. A search is an iterative optimizing process, and this process works well not only with static input but also with dynamic input. During any particular iteration, when the search detects that the problem input has changed, it adjusts a current solution to be consistent with the new constraints and then continues to improve this solution.

Considering the features described above, it is not difficult to see that Tabu Search offers an ideal framework for the Planning Agent to address the challenges listed earlier in Section A-1 of this Appendix. Specifically:

- Since Tabu Search is an incremental optimization method, at any point of time, there is always one current solution available. This current solution may be an empty solution, a partial solution, or a complete solution. In other words, if there is a solution to the problem, the search will find it. If there is a better solution, it will also find it, and if there is no solution, it will do its best to come up with a partial solution. Furthermore, because Tabu Search does not perform a complete search before obtaining a solution, it has a high probability of finding a feasible solution (if any exists) within a reasonable time frame. However, this presupposes that Tabu Search has been properly implemented so that none of the sub-steps, such as generate move, can have exponential time complexity.

- By design, Tabu Search always attempts to improve its current solution, regardless of whether the input to the problem is static or dynamic. Thus, when it detects changes to an input it does not completely discard the current solution, but adapts the current solution to the new circumstance. Such a strategy saves time in cases where the changes are not substantial (which applies to most situations in reality).

Figure A-3: Search flow for static input
Figure A-4: Search flow for dynamic input

---

22 It must be ensured that a step such as generate move is executed in polynomial time. The reason is that the key purpose of using the Tabu Search Framework is to find a solution for a NP problem within polynomial time, which can be guaranteed only if all sub-steps of Tabu Search can be done in polynomial time.
A-3 Design of the Planning Agent

In the TRANSWAY® application, a search process starts with the creation of a \textit{scenario} object. A \textit{scenario} object represents a problem instance and contains information that the Planning Agent needs to perform its functions (e.g., a topographic map that contains nodes and routes, a list of requests, and some available conveyances). A search process is iterative by design, and each iteration comprises four sequential steps: Create Move; Select Move; Apply Move; and, Update Solution. Figure A-3 depicts the general search flow of the Planning Agent when the problem input is static (i.e., the content of the \textit{scenario} object does not change). In cases where the problem input does change, the Planning Agent performs some additional operations to adapt the change (Figure A-4).

Object representations employed by the Planning Agent include:

- **Trip**: A Trip represents a possibility (i.e., topologically) that some supplies can be delivered from one location to another. A Trip is only a possibility because it has yet to be applied to constraint checking (e.g., conveyance availability, weight/space availability, and time constraints).

- **Scheduled Trip**: Once a Trip satisfies all constraints, it can be converted to a Scheduled Trip and become part of a found solution.

- **Operator**: An Operator represents an operation that can be applied to a Trip, so that the Trip can be part of a solution in the form of a Scheduled Trip. In the current version of TRANSWAY® four operators have been implemented, namely: \textit{add trip}; \textit{remove trip}; \textit{merge cargo}; and, \textit{shift time window}.

- **Move**: A Move represents a set of operations that can potentially improve the Current Solution. A Move is composed of a list of Operators.

- **Current Solution**: The Current Solution consists of a list of Scheduled Trips (i.e., zero or more). It represents a solution that the Planning Agent has found at a particular time. The Current Solution evolves as the Planning Agent tries to improve it.

- **Best Solution**: It is the best solution that the Planning Agent has found at a particular time. At the end of an iterative search cycle, Current Solution is compared with Best Solution. If Current Solution is better, then it is copied and becomes the new Best Solution.

- **Tabu List**: The Tabu List holds a list of moves that have been applied previously. The purpose of such a list is to force the search to select a move that has not been already explored (i.e., non-Tabu move). It provides an adaptive memory that allows the algorithm to remember where it has already been.

A-3.1 Create Moves

The purpose of Create Moves is to generate a list of possible moves that can satisfy a set of constraints introduced by some delivery requests. As illustrated in the Figure A-5, two main procedures are involved: Trip Generator; and, Move Generator. Trip Generator computes a set of possible Trips, based on the existence of some requests, a topographic map, and the availability of conveyances. It works in a way that is similar to solving the classic Finding Shortest-Path problem, except that it has more constraints to consider. It searches for not only
the shortest paths but also other paths, since a shortest path is not guaranteed to be a feasible path in the given context. Move Generator, on the other hand, computes a set of possible Moves, aiming to improve the Current Solution. A Move represents some operations that can modify the Current Solution, such as adding a trip, removing a trip, merging trips, and changing time windows.

![Data flow diagram of Create Moves](image)

Both procedures (i.e., Trip Generator and Move Generator) are computationally intensive and algorithmically complex. Their implementations are introduced in Section A-3.2, below.

### A-3.2 Select Move

The purpose of Select Move is to choose the best move among all the moves generated in the previous step (i.e., Create Move). Usually, a move is considered to be Best Move if: (1) it is not in the Tabu List; and, (2) it has the best objective values as compared with the previous moves (Figure A-6). A Tabu List is a memory structure that records the moves that have been applied previously and whose function is to drive the search towards some unexplored solution space. Objective values of a move are computed based on some subjective evaluation criterion that is usually user-defined (e.g., total shortfall or total travel distance).

The strategy used by Select Move determines the direction of a search flow and the search space that is eventually covered by the algorithm. Therefore, how the algorithm is implemented has substantial impact on its effectiveness in terms of how quickly a solution can be found and the quality of the solution. The current implementation of Select Move is shown in Figure A-6.
A-3.3 Apply Move

Once a move has been selected by Select Move, it needs to be applied to Current Solution so that the Current Solution can be modified. Specifically, the Current Solution is re-evaluated after the move has been applied to determine whether or not it has been improved. As mentioned previously, a move consists of a list of Operators, and an Operator can represent any one operation such as adding a Trip, merging a Trip, removing a Trip, or shifting the time window of a Trip. When applying a move to the Current Solution, two procedures are executed: Update Tabu List; and, Apply Operators (Figure A-7). Update Tabu List places the move on the Tabu List, so that this particular move will not be selected in the near future for another search. Apply Operators performs any necessary operations defined by the move to modify the Current Solution.
A-3.4 Update Solution

Update Solution determines whether the new Current Solution, created in the previous step, is better than the Best Solution found thus far. The comparison is made by looking at the objective values of the two solutions. If the Current Solution is indeed better than the Best Solution, then a clone of the Current Solution becomes the new Best Solution (Figure A-8). Update Solution is the last step of a Tabu Search iteration. After this step, a new iteration will start if the termination condition of the search has not been met.

![Data flow diagram of Update Solution](image)

Figure A-8: Data flow diagram of Update Solution

A-4 Implementation of the Planning Agent

This section describes how the design of the Planning Agent (as discussed in Section A-3) has been implemented. For each topic, the discussion starts with a high-level description of the component(s), followed by the pseudo code (derived from the actual Java code) for one or more principal algorithms, and concludes with some analysis.

A-4.1. Tabu Search Framework (OpenTS)

The Planning Agent is an adaptation of OpenTS, a Java implementation of Tabu Search Framework by Robert Harder (2001), to solve a dynamic vehicle scheduling and routing problem. Since the general search flow and control mechanisms are already provided in OpenTS, the main remaining tasks were to implement the four iterative steps (i.e., Create Moves, Select Move, Apply Move, and Update Solution) to target the specific problem domain that the agent is required to solve.

The general search flow and control mechanism of Tabu Search are implemented in a function called `performOneIteration()`, (i.e., source file: `SingleThreadedTabuSearch.java` under package OpenTS). Figure A-9 shows the pseudo code for this function.

The four steps of a search iteration can be easily identified in the pseudo code, which is designed to conduct Tabu Search in a generic fashion. However, to customize this method for a specific problem domain is dependent on the implementation of the four search steps, namely: Create Moves; Select Move; Apply Move; and, Update Solution.
A-4.2 Create Moves

It is the function of Create Moves to compute some possible moves that can potentially improve the Current Solution. Most of the operations of this step are accomplished by a function called `getAllMoves()` (source file: `TabuyaMoveManager.java` under package `planning`). Figure A-10 shows the pseudo code for this function.
As the pseudo code shows, `getAllMoves()` first generates a list of Trips based on each Request in the `scenario` object by calling `getTripForRequest()` and it then uses these Trips to create a list of moves by calling `createMoves()`. Keeping in mind that a move is created to improve the Current Solution, when there is no feasible move the algorithm creates a special move that causes the removal of a scheduled trip from the Current Solution. Generally speaking, this removal operation allows the search to explore a different solution space in the next iteration when that Current Solution space has been fully explored. Other strategies may include allowing the removal of a scheduled trip and the addition of a different scheduled trip for the same move. This could be more advantageous than a random removal since it would identify which Scheduled Trip would have to be removed in order to add a different one, allowing a more methodical exploration of the solution space.

There are two main functions that `getAllMoves()` is dependant on:

- `getTripForRequest()` (source file: `TripGenerator.java` under package `trip`), and
- `createMoves()` (source file: `OperatorService.java` under package `operatorservice`).

### A-4.2.1 Function `getTripForRequest()`

The pseudo code for the function `getTripForRequest()` is shown in Figure A-11. It calls for another function, `createRequestTrips()`, to compute a list of possible raw Trips for each Request, and then filters out those Trips that cannot meet the time windows of the Request that they are associated with. It is worth noting that even though the Trips have been checked against some constraints, other constraints have not yet been applied (e.g., exact scheduling, weight, and space). Another factor is the change of representation that the Trip object undergoes within this function. When a Trip is computed by function `createRequestTrips()` it is represented as a linked-list in which each element of the list is a trip-segment. However, the linked-list is changed to an array<Trip> representation.

![Figure A-11: Pseudo code for `getTripsForRequest()`](image_url)

23 A raw Trip is a Trip that has been created based on the topographic map alone and has yet been checked for other constraints.
As mentioned earlier, the generation of raw Trips is accomplished by calling function `createRequestTrips()` (source: `RequestTrips.java` under package `trip`), and `createRequestTrips()` relies on function `getTrip()` (source: `RequestTrips.java` under package `trip`).

### A-4.2.2 Function `createRequestTrip()` and `getTrip()`

The pseudo code for `createRequestTrip()` is shown in Figure A-12. This function finds raw trips that may satisfy each request, but the real work is accomplished by calling function `getTrips()`. The latter is among the most computationally intensive functions that has been implemented in the Planning Agent. Similar to other functions, `getTrips()` relies on a set of other helper functions to accomplish its task. The interaction between these helper functions and `getTrips()` is illustrated in Figure A-13, and the pseudo code for `getTrips()` and its helper functions is shown in Figures A-14, A-15, A-16, and A-17.

![Figure A-12: Pseudo code for `createRequestTrip()`](image)

![Figure A-13: Interactions between `getTrips()` and its helper functions](image)

What `getTrips()` tries to overcome is a difficult problem. In addition to solving the shortest-path problem, `getTrips()` also computes and stores other possible paths. Therefore, in the worst case, its time complexity increases exponentially as the number of nodes and routes increases. At this point, two strategies are implemented to prevent `getTrips()` from becoming a performance bottleneck: (1) save all the Trips that `getTrips()` generates into cache memory for future reference; and, (2) limit the total number of Trips it has to create for each Request. The first strategy does not save time when the Planning Agent enters the first iteration. However, for any subsequent iteration the Planning Agent will look up only the cache memory to fetch corresponding Trips, unless the Request has changed since the last iteration. The second strategy saves time by preventing `getTrips()` from going through every combination of routes to find possible Trips.
Unfortunately, there is a penalty associated with this strategy. Some solution spaces that could potentially contain a better solution will not be explored. To overcome this limitation the number of Trips that are considered can be incrementally updated as the search runs for an extended period of time. This strategy assumes that the majority of the solutions will be found with a relatively small number of raw Trip options for each request and covers the case where more trips are needed to find the optimum solution.

---

**Figure A-14: Pseudo code for getTrips()**

```java
PROCEDURE: getTrips(requestTrips, requestLocation, supplyLocations, prevCDR) --> requestTrips.java
INPUT: see signature above
OUTPUT: a list of Trips

// FORWARD SEARCH
1. create all possible trips can reach RL from SL directly by one segment; // calling GetCSRs();
   // try to connect segments
2. FOR (each trip)
3.   IF (prevCDR != null)
4.     THEN connect this trip to prevCDR;
5.     IF (this is NOT a valid trip)   // by calling ValidDrop()
6.       goto step 11;
7.     END IF;
8.   END IF;
9.   // if any, then add it into output tripList
10.  Add this trip to tripList;
11. END FOR;

// BACKWARD SEARCH
12. create all possible trips that can reach RL by one segment from nearby locations // calling getCDRs();
13. // try to connect segments
14. FOR (each trip)
15.   IF (prevCDR != null)
16.     THEN connect this trip to prevCDR
17.     IF (this is NOT a valid trip)
18.       goto step 21;
19.     END IF;
20.   END IF;
21.   // if any such trip, add it into expandable tripList
22.   add this trip to expandable list
23. END FOR

// EXPAND TRIP
24. FOR (each trip in the expandable list)
25.   let its SP be the RL, recursively call getTrip();
26. END FOR;
```

---

**Figure A-15: Pseudo code for helper function getCSRs()**

```java
PROCEDURE: getCSRs(supplyLocations, requestLocation)
INPUT: see signature above
OUTPUT: a list of trips // ArrayList<Trip>

1. FOR each location that has available vehicles
2.   FOR (each location that has supplies requested)
3.     get the Supply from the location, save it into currentSupply;   // Class Supply
4.     FOR (each available vehicle_type at that location)
5.       IF (the vehicle type can go to supply location) AND (from supply location go to request location)
6.         THEN (create a new trip) AND (add this trip to the trip list);
7.         END IF;
8.     END FOR;
9.   END FOR;
10. END FOR;
```
A-4.2.3 Function createMoves()

As shown in Figure A-10, besides getTripForRequest(), another important function that the search step Create Moves relies on is createMoves().

```
PROCEDURE: createMoves() -> OperatorService.java
INPUT: a solution, a trip with all segments: <ArrayList<Trip>>
OUTPUT: a list of tabuyasMove: <ArrayList<tabuyasMove>>
1. IF (trip list is empty) OR (request is no longer supplantable)
2. THEN return empty move list, DONE;
3. END IF;
4. create new operatorSet for this trip, put it into a list; //ArrayList<OperatorSet>
5. FOR (each trip segment)
6. try define operatorSets_M to merge it into some scheduled segments //calling findOverLaps()
7. try define operatorSets_N to convert it into some scheduled segments
8. END FOR;
9. put operatorSets_M & operatorSets_N together,
10. FOR (each operatorSet)
   //e.g.: AddCargoOperator, AddTripsOperator, AdjustWindowOperator
   create a list of operators; //createOperators(solution)
   create a TabuyasMove containing the list of operators, put the move into a moveList;
11. END FOR;
12. return moveList;
```
The pseudo code for `createMoves()` (source code: `OperatorService.java` under package `operatorservice`) is shown in Figure A-18. The purpose of `createMove()` is to improve the Current Solution. It achieves this by invoking one of the following strategies:

- Merging a Trip into the Current Solution.
- Adding a Trip to the Current Solution.
- Removing a Scheduled Trip from the Current Solution.
- Shifting the time window of a Scheduled Trip in the Current Solution.

Each of the above operations is performed by an Operator, and a set of possible Operators constitute a move. One function that `createMove()` relies on to define a merging operation is `findOverLaps()` (source code: `OperatorService.java` under package `operatorservice`), whose pseudo code is given in Figure A-19.

```
PROCEDURE: findOverLaps(thisSegment, solution, currentOps) -> OperatorService.java
INPUT: a solution, a trip segment, operator list
OUTPUT: operator list: ArrayList<OperatorSet>

0. Create an operator list; // ArrayList<OperatorSet>
1. FOR (each scheduledSegment in the solution)
2.     IF (scheduledSegment and thisSegment have same SL & RL)
3.         THEN IF (thisSegment can merge into scheduledSegment) //check weight&space availability
4.             THEN create a new operator does the merge, add it to the return list;
5.         END IF;
6.     END IF;
7. END FOR;
8. return resulted operator list;
```

Figure A-19: Pseudo code for `findOverLaps()`

As shown in Figure A-18, the current implementation of `createMove()` is based on checking every candidate Trip against all Scheduled Trips in the Current Solution, which can be potentially expensive as the total number of either candidate Trips or Scheduled Trips increases.

### A-4.3 Select Move

The pseudo code for this step is shown in Figure A-20. Two main operations are involved: (1) determining whether or not a move is in the Tabu List; and, (2) determining which move is the best among all candidates. Finding out if a move is in Tabu List or not is accomplished by calling function `isTabu()` (source code: `TabuyaList.java` under package `planning`), and the pseudo code of `isTabu()` is given in Figure A-21. As shown in Figure A-21, the current implementation relies on: (1) the hash code of a move; and, (2) a random guess (with a certain probability) to determine if a move is in the Tabu List. Further improvement of this method is currently under development.

---

24 Since the implementation of an Operator is straight-forward, the implementation details are not given here. They can be found under package `planning.operators`. 
Determining which move is best is accomplished by comparing the objective values of all candidate moves. The objective values of a move is obtained by calling function `evaluateMove()` (source code: `TabuyaObjectiveFunction.java` under package `planning.operators`). The pseudo code for `evaluateMove()` is give in Figure A-22.

```
PROCEDURE: evaluateMove() --> TabuyaObjectiveFunction.java
INPUT: solution:Solution
      move:Move
OUTPUT: double[] // objective values

//1st iteration, obValues were computed during initialize() -->BaseSolution.java
1. get objective values from the solution that Move will apply on:
   - go through all remaining Requests, get total shortfall;
   - go through all Scheduled Trips, get total distance;
2. get objective values assume Move is applied to solution;
3. add them all together;
4. return the two Numbers;
```

Figure A-20: Pseudo code for `getBestMove()`

```
PROCEDURE: isTabu() --> TabuyaList.java
INPUT: solution:Solution
       move:Move
OUTPUT: boolean

// this part needs to be further developed...
1. check short-term-memory (for the move only)
   - Add hashCode of all operators of the move;
   - look it up in the TabuList;
   - also do a random guess; // probability 0.8
   - if all of above are TRUE, return TRUE;
2. check long-term-memory
   - add up tabu values of solution & move
   - do a random guess; //again, given certain probability
3. return result;
```

Figure A-21: Pseudo code for `isTabu()`

```
PROCEDURE: evaluateMove() --> SingleThreadedTabuSearch.java
INPUT: solution:Solution
       moves:Move[
         objectiveFunction:ObjectiveFunction
         tabulist:TabuList
         aspirationCriteria:AspirationCriteria
         maximizing:boolean
         chooseFirstImprovingMove:boolean
       ]
OUTPUT: bestMove, objectiveValue[],
       tabuStatus

1. set the 1st move as the BestMove;
2. evaluate this move; //calling evaluateMove(solu, move)
3. find out if it is Tabu; //calling isTabu()
4. going through all Moves to pick the best move;
5. return {bestMove, values[], true/false}
```

Figure A-22: Pseudo code for `evaluateMove()`
Currently, two objective values are used: total shortfall, and total distance. Potentially, other types of evaluation criteria, such as monetary cost (of all trips), potential threat of routes, etc., could also be taken into account. However, what criteria to use should be dependent on the users’ preferences.

**A-4.4 Apply Move**

From Figure A-9, it is easy to see that Apply Move performs two main operations: (1) applying a chosen move (i.e., the best move) to the Current Solution; and, (2) updating the objective values of the Current Solution. The first operation is accomplished by calling function `TabuyaMove.operateOn()` (source code: `TabuyaMove.java` under package `planning.operators`). The pseudo code for `operateOn()` is shown in Figure A-23.

```
PROCEDURE: operateOn() -> TabuyaMove.java
INPUT: Move, Current Solution
OUTPUT: void //new Current solution
1. FOR (each Operators in the Move)
2. apply Operator to Solution;
3. END FOR;
4. perform operations to maintain dependencies;
```

Figure A-23: Pseudo code for `operateOn()`

As mentioned earlier, applying an Operator updates the Current Solution by adding a Trip, merging two Trips (i.e., adding cargo to an existing Trip), removing a Trip, or shifting the time window for a Trip. The updating of the objective values of the Current Solution is accomplished by recalculating the total shortfall and total traversal distance of the Current Solution.

**A-4.5 Update Solution**

The pseudo code of this step is shown in Figure A-24, which is part of the function `performOneIteration()` (Figure A-9).

```
PROCEDURE: Update Solution -> part of performOneIteration()
INPUT: old Current Solution
       new Current Solution
       Best Solution
OUTPUT: Best Solution
1. IF (currentSolution is improved) AND (currentSolution is better than bestSolution)
2. THEN set currentSolution as bestSolution;
3. END IF;
```

Figure A-24: Pseudo code for Update Solution

Although Update Solution appears to be the simplest among all four steps of a Tabu Search interaction, its significance should not be overlooked. Recall that Tabu Search is an incremental optimization method. Therefore, whether or not a solution has really been improved is dependent on answers to the following two questions: How can we find a move that can potentially improve a solution? How do we know if a solution has been improved?
The Planning Agent answers the first question by generating moves that can modify the Current Solution (i.e., through adding a Trip, merging Trips, etc.). Nevertheless, did the move really improve a solution? This is determined by comparing the new Current Solution to the Best Solution at the Update Solution step. Evaluation criteria used for performing such comparisons are crucial to the success of this algorithm. Currently, two solutions are compared by examining two objective values: total shortfall; and, total traversal distance. The Tabu List also adds a random guess element (with a given probability) that may deter the search from optimal solutions. More accurate comparison methods are still under development.

A-5 Performance Analysis

The principal advantage of using the Tabu Search Framework is that it is designed for finding a feasible (i.e., a good enough) solution within a reasonable time frame. Therefore, none of its internal operations should take exponential time to complete. However, the time complexity of Create Moves can grow exponentially as the number of nodes and routes increases. As previously mentioned, two strategies have been implemented for improvement: (1) a cache memory structure is used to store all of the raw trips generated by the Trip Generator during the first iteration of the algorithm, so that subsequent iterations will not have to recompute these trips; and, (2) for each Request, the total number of Trips generated by the Trip Generator is bound by a constant $K$, so that not all possible Trips are considered.

A set of tests have been devised to evaluate the performance of the Planning Agent based on the current implementation. These tests are designed to answer the following questions:

1. What is the time distribution among the four steps (i.e., Create Moves, Select Move, Apply Move, and Update Solution) of one search iteration?

2. Are there any bottleneck and, if yes, how can they be improved or eliminated?

The same static scenario was used for all tests. It included: 174 Nodes (i.e., Locations), 212 Routes, 258 Conveyances, 10 Requests, and 14 Supplies. However, for each test the value of $K$ (i.e., the upper bound for the total number of trips created by Trip Generator) was changed. Also, each test was terminated after five search iterations, with the time recorded for the four steps within each iteration.

The results clearly indicated that among the four steps (i.e., Create Move, Select Move, Apply Move, and Update Solution), Create Moves takes the longest time to complete. In fact, when compared to the time that Create Moves takes, the time taken by other steps is almost negligible (i.e., 686,366 msec v.s. 3.5 msec). Within the step of Create Moves, the first iteration takes the longest time, while the other iterations are essentially constant in time. Also, while the bound value $K$ increases (i.e., 2, 4, 8,… 512), the time taken by Create Moves increases almost linearly.

Since the first step of a Tabu Search iteration, Create Move, takes almost all the time needed to complete one iteration, Create Moves clearly constitutes a bottleneck to the overall system. The question then becomes: Which procedure is taking the most time within Create Move? As discussed previously, Create Moves contains two main functions, `getTripForRequest()` and `CreateMoves()`. The first function generates raw Trips, and the second function tries to create moves to improve the Current Solution. Since cache memory is used in `getTripForRequest()`, most of its computation is performed during the first iteration of the Search (subsequent iterations mostly just look-up Trips from in cache). Therefore, it can be concluded that
getTripForRequest() is the function that takes most of the time to complete. For example, the results of the first test indicated that the first iteration of Create Moves took 367,997 msec, while the second iteration took only 5 msec. Since we know that the function getTripForRequest() did not really execute during the second iteration, it is clear that CreateMoves() took 5 msec and getTripForRequest() took approximately 367,992 msec\(^{25}\).

More tests are expected to be performed in the future as the Planning Agent evolves, including tests that evaluate the performance of Planning Agent under dynamic input conditions.

A-6 Discussion

In summary, the main advantage that Tabu Search Framework has provided is the separation of the optimization process from the constraint verification process (Weber and Bojduj 2006). The optimization process in Tabu Search is implemented as an iterative search process, where each iteration seeks to improve the Current Solution by adding a Trip or modifying a Trip. The constraint verification process is accomplished mainly within the Create Moves step.

The current implementation of the Planning Agent appears promising and is encouraging. Nevertheless, as a guideline for future development the following areas are candidates for further research aimed at improving the performance of the Planning Agent:

- **Design and use of Tabu List:** Although this is a crucial component of the Tabu Search Framework, the current implementation has not yet captured its full potential. The following specific issues appear to be worthy of further attention: Could there be a better representation of a Move/Solution in Tabu List that will allow the comparison to be performed more accurately and efficiently? Would it be useful to apply the concept of short-term and long-term memory?

- **Comparison method for selecting a move and choosing a solution:** This has much to do with the implementation of the objective function and, specifically the criteria used by the objective function to evaluate a move and/or a solution. As discussed previously, Tabu Search relies heavily on these criteria to accomplish what it is designed for. In other words, how the objective function is used in comparison has significant impact on the effectiveness of the overall performance of Tabu Search and the quality of the generated solution.

- **The implementation of Create Moves:** Specifically, the current implementation of getTripForRequest() is the main focus from a performance standpoint. Future research should include looking for search techniques and/or utilizing some heuristics/constraints to narrow the search spaces more efficiently. New requirements such as allowing a vehicle to perform multiple pick-ups and deliveries will require the getTripForRequest() algorithm to be re-written. These requirements will undoubtedly increase the complexity of this algorithm making the above mentioned concern ever more important and pressing. A brute force approach is unlikely to provide a suitable solution even if bounded like previous approaches. Instead, an incremental approach that executes asynchronously may be required.

\(^{25}\) 367,992 = (367,997 – 5)
• **Distributed computing:** A substantial gain in performance may be achievable with a distributed-computing approach.
Appendix B: The TRANSWAY® Data Dictionary

This Appendix contains the TRANSWAY® data dictionary that applies to the Domain Model diagrams and ontology description provided in Chapter Four. Each domain (i.e., package) within the Domain Model is given below as a section and each domain section is divided into subsections relating to its important parts (i.e., structs, classes, and enumerations). Within each subsection, information is provided about attributes, roles, and inheritance.

Package: transway

Enumeration : eRouteType

Allowed Values:

AIR_RT PAVED_ROAD_RT UNPAVED_ROAD_RT WATERWAY_RT RAIL_RT

Struct : sPosition

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>longitude (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>altitude (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

Struct : sObjectReference

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>refObjectKey</td>
<td>long</td>
<td>read-write</td>
<td>documentation: Object Key of the object being referenced.</td>
</tr>
<tr>
<td>className</td>
<td>string</td>
<td>read-write</td>
<td>documentation: Full path of class name of object being referenced. E.g., com.cdmtech.core.hibernate.transway.environment.SupplyCenter.</td>
</tr>
</tbody>
</table>

Struct : sDimensions

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>length (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>width (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>height (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

26 A struct allows a number of attributes to be grouped together, using accessor methods, without having to write an explicit class.
Class : **TranswayObject**

Direct Subclasses:
- TimeWindow
- SubRequirement
- AttributeInformation
- Impediment
- Transfer
- AreaToIgnore
- Constraint
- Clock
- RouteSegment
- Note
- ObjectiveConstraint
- EmbeddedConstraint
- ClassInformation
- Alert
- Waypoint
- OrganizationEntity
- CargoTransfer
- Requirement
- AreaOfInterest
- Item
- Solution
- TranswayType

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

Class : **AttributeInformation**

Extends:
- TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>className</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation</strong>: Name of the class where the attribute is defined.</td>
</tr>
<tr>
<td>displayName</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation</strong>: Name to be displayed to the user for an attribute.</td>
</tr>
<tr>
<td>unitStored (NONE_MU)</td>
<td>eMeasurementUnit</td>
<td>read-write</td>
<td><strong>documentation</strong>: Measurement unit stored for the attribute.</td>
</tr>
<tr>
<td>isPropertyConstraint</td>
<td>boolean</td>
<td>read-write</td>
<td><strong>documentation</strong>: Defined as true if the attribute is a PropertyConstraint rather than a standard attribute.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
- name
- description

Enumeration : **eMeasurementUnit (AttributeInformation)**

Allowed Values:

<table>
<thead>
<tr>
<th>NONE_MU</th>
<th>DISTANCE_INCHES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISTANCE_FEET</td>
<td>DISTANCE_YARDS</td>
</tr>
<tr>
<td>DISTANCE_MILES</td>
<td>DISTANCE_POINTS</td>
</tr>
<tr>
<td>DISTANCE_MILLIMETERS</td>
<td>DISTANCE CENTIMETERS</td>
</tr>
<tr>
<td>DISTANCE_METERS</td>
<td>DISTANCE KILOMETERS</td>
</tr>
<tr>
<td>DISTANCE NAUTICAL MILES</td>
<td>TIME_MILLISECONDS</td>
</tr>
<tr>
<td>TIME SECONDS</td>
<td>TIME_MINUTES</td>
</tr>
<tr>
<td>TIME HOURS</td>
<td>TIME_DAYS</td>
</tr>
<tr>
<td>VELOCITY MILES_PER_HOUR</td>
<td>VELOCITY METERS PER SECOND</td>
</tr>
<tr>
<td>VELOCITY_KM_PER_HOUR</td>
<td>VELOCITY KNOTS</td>
</tr>
</tbody>
</table>
**Class : ClassInformation**

Extends: 
TranswayObject
Attributes inherited from TranswayObject:
name description

**Class : TranswayType**

Extends: 
TranswayObject
Direct Subclasses:
DisplayAgentType OrganizationType AlertTypeItemType InternalAgentType ConveyanceLayout PersonType
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>customType (false)</td>
<td>boolean</td>
<td>read-write</td>
<td>documentation: Defined as true if the object is created custom by the user and false otherwise.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
name description

**Package: time**

**Class : Clock (time)**

Extends: 
TranswayObject
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>status (PAUSED_CS)</td>
<td>eClockStatus</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>time (0)</td>
<td>long</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>temporalFactor (0.0)</td>
<td>float</td>
<td>read-write</td>
<td>documentation: Rate at which the clock operates.</td>
</tr>
<tr>
<td>timeBetweenUpdates (0)</td>
<td>long</td>
<td>read-write</td>
<td>documentation: Rate at which the clock updates.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
name description

**Enumeration : eClockStatus (time.Clock)**

Allowed Values:
- PAUSED_CS
- EXECUTING_CS

**Class : TimeWindow (time)**

Extends:
- TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime (0)</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Absolute start time of the window.</td>
</tr>
<tr>
<td>endTime (0)</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Absolute end time of the window.</td>
</tr>
<tr>
<td>eventDuration (0)</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Active duration of the window.</td>
</tr>
<tr>
<td>eventFrequency</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Frequency within the window which the duration is active.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:

name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>availablePerson</td>
<td>Person</td>
<td>availablePerson availabilities</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>availableSupplyCenter</td>
<td>SupplyCenter</td>
<td>availableSupplyCenter availabilities</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>unavailablePerson</td>
<td>Person</td>
<td>unavailablePerson unavailable</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>conveyance</td>
<td>Conveyance</td>
<td>conveyance unavailabilities</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>impediment</td>
<td>Impediment</td>
<td>impedimenttimeWindow</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>unavailableSupplyCenter</td>
<td>SupplyCenter</td>
<td>unavailableSupplyCenter unavailable</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Package: types**

**Class : PersonType (types)**

Extends:
- TranswayType

Attributes inherited from TranswayType:
customType
Attributes inherited from TranswayObject:
  name description
PropertyConstraints:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDurationBeforeRest</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation:</strong> Amount of time that the person can work before a rest period is required.</td>
</tr>
<tr>
<td>restDuration</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation:</strong> Required rest duration for the person after a work period.</td>
</tr>
<tr>
<td>waitTimeBeforePickup</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation:</strong> Amount of time that the person can wait before being picked up at some location.</td>
</tr>
</tbody>
</table>

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyConstraints</td>
<td>PropertyConstraint</td>
<td>personTypepropertyConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

**Class : AlertType (types)**

Extends:
  TranswayType
Attributes inherited from TranswayType:
  customType
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalAgentType</td>
<td>InternalAgentType</td>
<td>alertTypes internalAgentType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>displayAgentType</td>
<td>DisplayAgentType</td>
<td>alertTypes displayAgentType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>violatedConstraints</td>
<td>EmbeddedConstraint</td>
<td>violatingAlertTypes violatedConstraints</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Class : ItemType (types)**

Extends:
  TranswayType
Direct Subclasses:
  CargoType ContainerType ConveyanceType
Attributes:
Attributes inherited from TranswayType:  
customType
Attributes inherited from TranswayObject:  
name description
Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyConstraints</td>
<td>PropertyConstraint</td>
<td>itemTypepropertyConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

**Class : CargoType (types)**

Extends:  
ItemType
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSN</td>
<td>string</td>
<td>read-write</td>
<td>documentation: National Stock Number of the cargo type.</td>
</tr>
<tr>
<td>supplyClass (I_SC)</td>
<td>eSupplyClass</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>commodityCode</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>quantityPerGroup (0)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Quantity of cargo within the cargo type. E.g., 30 bullets in a box of bullets.</td>
</tr>
</tbody>
</table>

Attributes inherited from ItemType:  
model weight dimensions
Attributes inherited from TranswayType:  
customType
Attributes inherited from TranswayObject:  
name description
PropertyConstraints:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>waitTimeBeforePickup</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td>documentation: Amount of time that the cargo type can wait before being picked up at some location.</td>
</tr>
</tbody>
</table>

Roles inherited from ItemType:
propertyConstraints
**Enumeration : eSupplyClass (types.CargoType)**

Allowed Values:
- I_SC: I
- II_SC: II
- III_SC: III
- IV_SC: IV
- V_SC: V
- VI_SC: VI
- VII_SC: VII
- VIII_SC: VIII
- IX_SC: IX

**Class : ContainerType (types)**

Extends:
- ItemType

Attributes inherited from ItemType:
- model
- weight
- dimensions

Attributes inherited from TranswayType:
- customType

Attributes inherited from TranswayObject:
- name
- description

Roles inherited from ItemType:
- propertyConstraints

**Class : ConveyanceType (types)**

Extends:
- ItemType

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>routeTypes</td>
<td>eRouteType</td>
<td>read-write</td>
<td><strong>documentation</strong>: Type of routes that the conveyance type can traverse.</td>
</tr>
<tr>
<td>milStdCode</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation</strong>: Military standard code (2525b) representing the conveyance type’s icon.</td>
</tr>
</tbody>
</table>

Attributes inherited from ItemType:
- model
- weight
- dimensions

Attributes inherited from TranswayType:
- customType

Attributes inherited from TranswayObject:
- name
- description

PropertyConstraints:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Distance of travel before refueling is necessary for the conveyance type.</td>
</tr>
<tr>
<td>cruisingSpeed</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Speed of travel given normal conditions for the conveyance type.</td>
</tr>
<tr>
<td>loadTime</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Amount of time it takes to load cargo and/or</td>
</tr>
</tbody>
</table>
unloadTime | PropertyConstraint | read-write | documentation: Amount of time it takes to unload cargo and/or passengers from the conveyance type.
--- | --- | --- | ---
costPerHour | PropertyConstraint | read-write | documentation: Cost of operation per hour of the conveyance type.
weightCapacity | PropertyConstraint | read-write | documentation: Allowed weight of cargo and/or passengers on the conveyance type.
timePerMission | PropertyConstraint | read-write | documentation: Amount of time allowed per mission for the conveyance type.
refuelTime | PropertyConstraint | read-write | documentation: Amount of time it takes to refuel the conveyance type.

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>layouts</td>
<td>ConveyanceLayout</td>
<td>layoutsconveyanceType</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

Roles inherited from ItemType:
propertyConstraints

Class: ConveyanceLayout (types)
Extends: TranswayType
Attributes inherited from TranswayType:
customType
Attributes inherited from TranswayObject:
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertyConstraints</td>
<td>PropertyConstraint</td>
<td>layoutpropertyConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>conveyanceType</td>
<td>ConveyanceType</td>
<td>layoutsconveyanceType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>containerType</td>
<td>ContainerType</td>
<td>layoutcontainerType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>itemConstraints</td>
<td>ItemConstraint</td>
<td>layoutitemConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>personConstraints</td>
<td>PersonConstraint</td>
<td>layoutpersonConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

Class: DisplayAgentType (types)
Extends: TranswayType
Attributes inherited from TranswayType:
customType
Attributes inherited from TranswayObject:
name description

**PropertyConstraints:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>passengerSpace</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Number of passengers that can be held in the conveyance layout.</td>
</tr>
<tr>
<td>crewSpace</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Number of crew members that can be held in the conveyance layout.</td>
</tr>
<tr>
<td>containerSpace</td>
<td>PropertyConstraint</td>
<td>read-write</td>
<td><strong>documentation</strong>: Number of containers that can be held in the conveyance layout.</td>
</tr>
</tbody>
</table>

**Roles:**

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertTypes</td>
<td>AlertType</td>
<td>alertTypes.displayAgentType</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Class : OrganizationType (types)**

Extends: 
TranswayType

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>echelon</td>
<td>eEchelonType</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

(UNKNOWN_ET)

Attributes inherited from TranswayType:

customType

Attributes inherited from TranswayObject:

customType

echelonName
description

**Enumeration : eEchelonType (types.OrganizationType)**

Allowed Values:

**Class : InternalAgentType (types)**

Extends: 
TranswayType

Attributes inherited from TranswayType:

customType

Attributes inherited from TranswayObject:

customType
echelonName
description

Roles:
### Package: constraints

**Class : ObjectiveConstraint (constraints)**

Extends:  
TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>consider (true)</td>
<td>boolean</td>
<td>read-write</td>
<td>documentation: Defined as true if the objective constraint should be considered and false otherwise.</td>
</tr>
<tr>
<td>index (1)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Numeric index of the objective constraint, used in comparison with other objective constraints.</td>
</tr>
<tr>
<td>rank (1)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Numeric rank of the objective constraint, used in comparison with other objective constraints.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:  
name description

**Class : EmbeddedConstraint (constraints)**

Extends:  
TranswayObject

Attributes inherited from TranswayObject:  
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>violatingAlertTypes</td>
<td>AlertType</td>
<td>violatingAlertTypesviolatedConstraints</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Class : Constraint (constraints)**

Extends:  
TranswayObject

Direct Subclasses:
PersonConstraint PropertyConstraint ItemConstraint

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
</table>
comparator (EQUAL_TO) | eComparator | read-write | **documentation**: Comparator for the constraint. E.g., less than, greater than.
---|---|---|---
value (0.0) | double | read-write | **documentation**: Value of the constraint to be compared with the comparator.

Attributes inherited from TranswayObject:
name description

**Enumeration : eComparator (constraints.Constraint)**

Allowed Values:

<table>
<thead>
<tr>
<th>EQUAL_TO</th>
<th>NOT_EQUAL_TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREATER_THAN_OR_EQUAL_TO</td>
<td>STRICTLY_GREATER_THAN</td>
</tr>
<tr>
<td>LESS_THAN_OR_EQUAL_TO</td>
<td>STRICTLY_LESS_THAN</td>
</tr>
</tbody>
</table>

**Class : PersonConstraint (constraints)**

Extends:
Constraint
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>height (-1.0)</td>
<td>double</td>
<td>read-write</td>
<td><strong>documentation</strong>: Height of a person represented by the person constraint.</td>
</tr>
<tr>
<td>weight (-1.0)</td>
<td>double</td>
<td>read-write</td>
<td><strong>documentation</strong>: Weight of a person represented by the person constraint.</td>
</tr>
</tbody>
</table>

Attributes inherited from Constraint:
comparator value
Attributes inherited from TranswayObject:
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>supplyCenter</td>
<td>SupplyCenter</td>
<td>supplyCenterpersonReserveConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>personType</td>
<td>PersonType</td>
<td>constraintpersonType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>layout</td>
<td>ConveyanceLayout</td>
<td>layoutpersonConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Class : PropertyConstraint (constraints)**

Extends:
Constraint
Attributes inherited from Constraint:
comparator value
Attributes inherited from TranswayObject:
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemType</td>
<td>ItemType</td>
<td>itemTypepropertyConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>personType</td>
<td>PersonType</td>
<td>personTypepropertyConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>layout</td>
<td>ConveyanceLayout</td>
<td>layoutpropertyConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Class : ItemConstraint (constraints)
Extends: Constraint
Attributes inherited from Constraint:
comparator value
Attributes inherited from TranswayObject:
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>ConveyanceLayout</td>
<td>layoutitemConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>itemType</td>
<td>ItemType</td>
<td>constraintitemType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>supplyCenter</td>
<td>SupplyCenter</td>
<td>supplyCenteritemReserveConstraints</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Package: items

Class : Item (items)
Extends: TranswayObject
Direct Subclasses: Container RolledUpCargo Conveyance
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>usage</td>
<td>eItemUsage</td>
<td>read-write</td>
<td>documentation: Specifies current usage of an item.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgEntity</td>
<td>OrganizationEntity</td>
<td>orgEntityitems</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Enumeration : eItemUsage (items.Item)
Allowed Values:
    EQUIPMENT_IU  SUPPLY_IU  AVAILABLE_IU  IN_TRANSIT_IU

Class : Container (items)
Extends:
  Item
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>sObjectReference</td>
<td>read-write</td>
<td>![documentation](Current location of the container (in reference to some physical location). E.g., a container is on some truck.)</td>
</tr>
<tr>
<td>ULN</td>
<td>string</td>
<td>read-write</td>
<td>![documentation](Unit Line Number of the container.)</td>
</tr>
<tr>
<td>TCN</td>
<td>string</td>
<td>read-write</td>
<td>![documentation](Transportation Control Number of the container.)</td>
</tr>
</tbody>
</table>

Attributes inherited from Item:
  usage
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cargo</td>
<td>RolledUpCargo</td>
<td>containercargo</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>containerType</td>
<td>ContainerType</td>
<td>containercargotype</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from Item:
  orgEntity

**Class: RolledUpCargo (items)**

Extends:
  Item
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantity (0)</td>
<td>int</td>
<td>read-write</td>
<td>![documentation](Quantity of cargo represented by the rolled up cargo object.)</td>
</tr>
</tbody>
</table>

Attributes inherited from Item:
  usage
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cargoType</td>
<td>CargoType</td>
<td>rolledUpCargocargoType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>container</td>
<td>Container</td>
<td>containercargo</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from Item:
  orgEntity
Class: Conveyance (items)

Extends: 
  Item
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>bhtNumber</td>
<td>string</td>
<td>read-write</td>
<td>documentation: Id number of the conveyance. E.g., bumper number of a truck.</td>
</tr>
<tr>
<td>position</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Current geospatial position of the conveyance.</td>
</tr>
</tbody>
</table>

Attributes inherited from Item:
  usage
Attributes inherited from TranswayObject:
  name description
Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>conveyanceType</td>
<td>ConveyanceType</td>
<td>conveyanceconveyanceType</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>unavailabilities</td>
<td>TimeWindow</td>
<td>conveyanceunavailabilities</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

Roles inherited from Item:
  orgEntity

Package: environment

Class: Impediment (environment)

Extends:
  TranswayObject
Direct Subclasses:
  WeatherImpediment OtherImpediment
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>perimeter</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Ordered geospatial perimeter of the impediment.</td>
</tr>
<tr>
<td>degree (LOW_De)</td>
<td>eDegree</td>
<td>read-write</td>
<td>documentation: Current degree of the impediment, determined by an agent.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
  name description
Roles:
<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Multi.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeWindow</td>
<td>TimeWindow</td>
<td>impediment</td>
<td>1...1</td>
<td>Aggregation</td>
</tr>
<tr>
<td>routeSegments</td>
<td>RouteSegment</td>
<td>routeSegmentsimpeiments</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Enumeration : eDegree (environment.Impediment)**

Allowed Values
- LOW_De
- MEDIUM_De
- HIGH_De

**Class : WeatherImpediment (environment)**

Extends:
- Impediment

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>precipitation (NONE_Pr)</td>
<td>ePrecipitation</td>
<td>read-write</td>
<td><strong>documentation</strong>: Precipitation of the weather impediment. E.g., rain, snow.</td>
</tr>
<tr>
<td>obstructions (NONE_Ob)</td>
<td>eObstructions</td>
<td>read-write</td>
<td><strong>documentation</strong>: Obstructions of the weather impediment. E.g., hail, sand.</td>
</tr>
<tr>
<td>speed (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>bearing (0.0)</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

Attributes inherited from Impediment:
- perimeter degree

Attributes inherited from TranswayObject:
- name
- description

Roles inherited from Impediment:
- timeWindow
- routeSegments

**Enumeration : eObstructions (environment.WeatherImpediment)**

Allowed Values:
- NONE_Ob
- HAIL_Ob
- SNOW_Ob

**Enumeration : ePrecipitation (environment.WeatherImpediment)**

Allowed Values:
- NONE_Pr
- LIGHT_Pr
- MODERATE_Pr
- HEAVY_Pr

**Class : OtherImpediment (environment)**

Extends:
- Impediment

Attributes:
<table>
<thead>
<tr>
<th>type</th>
<th>eImpedimentType</th>
<th>access</th>
<th>documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(UNKNOWN_IT)</td>
<td>read-write</td>
<td></td>
<td>Type of the impediment. E.g., explosion, unknown</td>
</tr>
</tbody>
</table>

Attributes inherited from Impediment:
- perimeter degree
Attributes inherited from TranswayObject:
- name description
Roles inherited from Impediment:
- timeWindow routeSegments

**Enumeration : eImpedimentType (environment.OtherImpediment)**

Allowed Values:
- UNKNOWN_IT
- ATTACK_IT
- EXPLOSION_IT

**Class : AreaOfInterest (environment)**

Extends:
- TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>perimeter</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Ordered geospatial perimeter of the area of interest.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
- name description

**Class : AreaToIgnore (environment)**

Extends:
- TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>perimeter</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Ordered geospatial perimeter of the area to ignore.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
- name description

**Class : RouteSegment (environment)**

Extends:
- TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>eRouteType</td>
<td>read-write</td>
<td>documentation: Type of route. E.g., paved road, waterway.</td>
</tr>
<tr>
<td>(PAVED_ROAD_RT)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### intermediateVertices

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPosition</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Geospatial position of the note.</td>
</tr>
</tbody>
</table>

**documentation:** Ordered geospatial vertices that exist between endA and endB.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>double</td>
<td>read-write</td>
<td>documentation: Current length of the route segment.</td>
</tr>
</tbody>
</table>

### Roles

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>impediments</td>
<td>Impediment</td>
<td>routeSegmentsimpediments</td>
<td>0...*</td>
<td>Association</td>
</tr>
<tr>
<td>endA</td>
<td>Waypoint</td>
<td>endArouteSegmentsA</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>endB</td>
<td>Waypoint</td>
<td>endBrouteSegmentsB</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Class: **Note (environment)**

Extends: TranswayObject

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Geospatial position of the note.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:

name description

Class: **Waypoint (environment)**

Extends: TranswayObject

Direct Subclasses:

SupplyCenter

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>sPosition</td>
<td>read-write</td>
<td>documentation: Geospatial position of the waypoint.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:

name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>routeSegmentsB</td>
<td>RouteSegment</td>
<td>endBrouteSegmentsB</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>routeSegmentsA</td>
<td>RouteSegment</td>
<td>endArouteSegmentsA</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>accommodates</td>
<td>ConveyanceType</td>
<td>waypointaccommodates</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

Class: **SupplyCenter (environment)**

Extends:
Waypoint

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEOLOC</td>
<td>string</td>
<td>read-write</td>
<td>documentation: Geographical Location (GEOLOC) or location code of the supply center.</td>
</tr>
<tr>
<td>ICAO</td>
<td>string</td>
<td>read-write</td>
<td>documentation: International Civil Aviation Organization (ICAO) code of the supply center.</td>
</tr>
<tr>
<td>IATA</td>
<td>string</td>
<td>read-write</td>
<td>documentation: International Air Transport Association (IATA) code of the supply center.</td>
</tr>
<tr>
<td>portCode</td>
<td>string</td>
<td>read-write</td>
<td>documentation:</td>
</tr>
<tr>
<td>MOGw (0)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Maximum On Ground Working at the supply center. I.e., maximum number of conveyances that can be loaded/unloaded at the supply center at any particular time.</td>
</tr>
<tr>
<td>MOGp (0)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Maximum On Ground Parking at the supply center. I.e., maximum number of conveyances that can be parked at the supply center at any particular time.</td>
</tr>
<tr>
<td>throughput (0.0)</td>
<td>double</td>
<td>read-write</td>
<td>documentation: Maximum stons/hour at the supply center.</td>
</tr>
<tr>
<td>weight (0.0)</td>
<td>double</td>
<td>read-write</td>
<td>documentation: Current weight of cargo at the supply center.</td>
</tr>
</tbody>
</table>

Attributes inherited from Waypoint:
- position

Attributes inherited from TranswayObject:
- name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemReserve Constraints</td>
<td>ItemConstraint</td>
<td>supplyCenter itemReserveConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>availabilities</td>
<td>TimeWindow</td>
<td>availableSupplyCenter availabilities</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>organizations</td>
<td>Organization</td>
<td>organizationshomeLocation</td>
<td>0...*</td>
<td>Association</td>
</tr>
<tr>
<td>personReserve Constraints</td>
<td>PersonConstraint</td>
<td>supplyCenter personReserveConstraints</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>unavailables</td>
<td>TimeWindow</td>
<td>unavailableSupplyCenter unavailables</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>
Roles inherited from Waypoint:
  routeSegmentsB routeSegmentsA accommodates

**Package: organizations**

**Class :** OrganizationEntity (organizations)
Extends:
  TranswayObject
Direct Subclasses:
  Detachment Person Organization
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DODAAC</td>
<td>string</td>
<td>read-write</td>
<td>documentation: Department Of Defense Activity Address Code (DODAAC) of the organization, person, or detachment.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject: name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgReq</td>
<td>OrganizationalRequirement</td>
<td>orgReqorgEntity</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>items</td>
<td>Item</td>
<td>orgEntityitems</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

**Class :** Detachment (organizations)
Extends:
  OrganizationEntity
Attributes inherited from OrganizationEntity:
  DODAAC
Attributes inherited from TranswayObject: name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentOrganization</td>
<td>Organization</td>
<td>childDetachmentsparentOrganization</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>personnel</td>
<td>Person</td>
<td>parentDetachmentpersonnel</td>
<td>0...*</td>
<td>Association</td>
</tr>
<tr>
<td>childOrganizations</td>
<td>Organization</td>
<td>parentDetachmentschildOrganizations</td>
<td>0...*</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from OrganizationEntity: orgReq items

**Class :** Person (organizations)
Extends:
  OrganizationEntity
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation:</strong> Social security number (SSN) of the person.</td>
</tr>
<tr>
<td>lastName</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>firstName</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>middleName</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>dateOfBirth</td>
<td>long</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>eGender</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>bloodType</td>
<td>eBloodType</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>double</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

Attributes inherited from OrganizationEntity:
- DODAAC

Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentDetachment</td>
<td>Detachment</td>
<td>parentDetachmentpersonnel</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>parentOrganization</td>
<td>Organization</td>
<td>parentOrganizationpersonnel</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>unavailabilities</td>
<td>TimeWindow</td>
<td>unavailablePersonunavailabilities</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>personTypes</td>
<td>PersonType</td>
<td>personpersonTypes</td>
<td>0...*</td>
<td>Association</td>
</tr>
<tr>
<td>availabilities</td>
<td>TimeWindow</td>
<td>availablePersonavailabilities</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
</tbody>
</table>

Roles inherited from OrganizationEntity:
- orgReq items

**Enumeration : eBloodType (organizations.Person)**

Allowed Values:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN_BT</td>
</tr>
<tr>
<td>APOS_BT</td>
</tr>
<tr>
<td>ANEG_BT</td>
</tr>
<tr>
<td>BPOS_BT</td>
</tr>
<tr>
<td>ABPOS_BT</td>
</tr>
<tr>
<td>ABNEG_BT</td>
</tr>
<tr>
<td>OPOS_BT</td>
</tr>
<tr>
<td>ONEG_BT</td>
</tr>
</tbody>
</table>

**Enumeration : eGender (organizations.Person)**

Allowed Values:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MALE_GDR</td>
</tr>
<tr>
<td>FEMALE_GDR</td>
</tr>
<tr>
<td>UNKNOWN_GDR</td>
</tr>
</tbody>
</table>

**Class : Organization (organizations)**
Extends:
  OrganizationEntity

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIC</td>
<td>string</td>
<td>read-write</td>
<td>documentation: Unit Identification Code (UIC) of the organization.</td>
</tr>
</tbody>
</table>

Attributes inherited from OrganizationEntity:
  DODAAC
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>personnel</td>
<td>Person</td>
<td>parentOrganizationpersonnel</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>childOrganizations</td>
<td>Organization</td>
<td>parentOrganizationchildOrganization</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>childDetachments</td>
<td>Detachment</td>
<td>childDetachmentsparentOrganization</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>organizationType</td>
<td>OrganizationType</td>
<td>organizationTypeorganization</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>parentOrganization</td>
<td>Organization</td>
<td>parentOrganizationchildOrganization</td>
<td>1...1</td>
<td>Association</td>
</tr>
<tr>
<td>requirements</td>
<td>Requirement</td>
<td>requesterrequirements</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>parentDetachment</td>
<td>Detachment</td>
<td>parentDetachmentschildOrganizations</td>
<td>1...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>homeLocation</td>
<td>SupplyCenter</td>
<td>organizationshomeLocation</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from OrganizationEntity:
  orgReq items

Package: requirements

Class : Requirement (requirements)
Extends:
  TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>approved (false)</td>
<td>boolean</td>
<td>read-write</td>
<td>documentation: Defined as true if the requirement has been approved by the user and false otherwise.</td>
</tr>
<tr>
<td>priority (NORMAL RP)</td>
<td>eReqPriority</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>ETD (0)</td>
<td>long</td>
<td>read-write</td>
<td>documentation: Earliest Time of Delivery (ETD) of the requirement.</td>
</tr>
<tr>
<td>LTD (0)</td>
<td>long</td>
<td>read-write</td>
<td>documentation: Latest Time of Delivery (LTD) of the</td>
</tr>
</tbody>
</table>
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>subReqs</td>
<td>SubRequirement</td>
<td>requirementsubReqs</td>
<td>0...*</td>
<td>Aggregation</td>
</tr>
<tr>
<td>requester</td>
<td>Organization</td>
<td>requesterrequirements</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Enumeration : eReqPriority (requirements.Requirement)**

Allowed Values:

LOW_RP NORMAL_RP HIGH_RP

**Class : SubRequirement (requirements)**

Extends:

  TranswayObject

Direct Subclasses:

  CargoRequirement OrganizationalRequirement

Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>requirement</td>
<td>Requirement</td>
<td>requirementsubReqs</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Class : CargoRequirement (requirements)**

Extends:

  SubRequirement

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantity (0)</td>
<td>int</td>
<td>read-write</td>
<td>documentation: Quantity of cargo being requested by the cargo requirement.</td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cargoType</td>
<td>CargoType</td>
<td>cargoReqcargoType</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from SubRequirement:
  Requirement

**Class : OrganizationalRequirement (requirements)**

Extends:
SubRequirement
Attributes inherited from TranswayObject:
name description
Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgEntity</td>
<td>OrganizationEntity</td>
<td>orgReqorgEntity</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Roles inherited from SubRequirement:
requirement

Package: planning

Class: CargoTransfer (planning)
Extends:
   TranswayObject
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>subRequirement</td>
<td>sObjectReference</td>
<td>read-write</td>
<td>documentation: Reference to the sub requirement being fulfilled by the cargo transfer. A reference is used for performance purposes.</td>
</tr>
<tr>
<td>items</td>
<td>sObjectReference</td>
<td>read-write</td>
<td>documentation: References to the items for the cargo transfer. References are used for performance purposes.</td>
</tr>
<tr>
<td>passengers</td>
<td>sObjectReference</td>
<td>read-write</td>
<td></td>
</tr>
</tbody>
</table>

Attributes inherited from TranswayObject:
name description
Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>transfer</td>
<td>Transfer</td>
<td>transfercargoTransfers</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

Class: Solution (planning)
Extends:
   TranswayObject
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>planID</td>
<td>long</td>
<td>read-write</td>
<td>documentation: Internal ID representing a plan.</td>
</tr>
<tr>
<td>exporting</td>
<td>boolean</td>
<td>read-write</td>
<td>documentation: Agent state variable defined as true if the agent is currently exporting and false</td>
</tr>
</tbody>
</table>

153
Attributes inherited from TranswayObject:
  name description

**Class**: Transfer (planning)

*Extends:*
  TranswayObject

*Attributes:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>missionID</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation</strong>: Internal ID representing the conveyance mission of the transfer.</td>
</tr>
<tr>
<td>planID</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Internal ID relating to the planID for a solution.</td>
</tr>
<tr>
<td>conveyanceGroupID</td>
<td>long</td>
<td>read-write</td>
<td><strong>documentation</strong>: Internal ID representing the convoy of the transfer.</td>
</tr>
<tr>
<td>valid (true)</td>
<td>boolean</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>startTime</td>
<td>long</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>endTime</td>
<td>long</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>type (IN_TRANSIT_TT)</td>
<td>eTransferType</td>
<td>read-write</td>
<td><strong>documentation</strong>: Type of the transfer. E.g., in transit, (un)load, refuel.</td>
</tr>
<tr>
<td>approved (false)</td>
<td>boolean</td>
<td>read-write</td>
<td><strong>documentation</strong>: Defined as true if the user selects to soft book (i.e., save) the transfer and false otherwise.</td>
</tr>
<tr>
<td>numOfItems</td>
<td>int</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>numOfPassengers</td>
<td>int</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>startLocation</td>
<td>sObjectReference</td>
<td>read-write</td>
<td><strong>documentation</strong>: Reference to the starting waypoint for the transfer. A reference is used for performance purposes.</td>
</tr>
<tr>
<td>traversal</td>
<td>sObjectReference</td>
<td>read-write</td>
<td><strong>documentation</strong>: Reference to the route segment for the transfer. A reference is used for performance purposes.</td>
</tr>
</tbody>
</table>
conveyance | sObjectReference | read-write | **documentation**: Reference to the conveyance used by the transfer. A reference is used for performance purposes.

layout | sObjectReference | read-write | **documentation**: Reference to the conveyance layout used by the conveyance for the transfer. A reference is used for performance purposes.

Attributes inherited from TranswayObject:

Name | With | Association | Mult. | Type
--- | --- | --- | --- | ---
cargoTransfers | CargoTransfer | transfercargoTransfers | 0...* | Aggregation
postTransfers | Transfer | preTransferspostTransfers | 0...* | Association
preTransfers | Transfer | preTransferspostTransfers | 0...* | Association

**Enumeration : eTransferType (planning.Transfer)**

Allowed Values

IN_TRANSIT_TT REFUEL_TT LOAD_TT UNLOAD_TT

**Package: alerts**

**Class : Alert (alerts)**

Extends:

TranswayObject

Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Tagged Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>sObjectReference</td>
<td>read-write</td>
<td><strong>documentation</strong>: References to the objects related to the alert. References are used because a TranswayObject cannot be related to directly.</td>
</tr>
<tr>
<td>priority (LOW_AP)</td>
<td>eAlertPriority</td>
<td>read-write</td>
<td><strong>documentation</strong>: Priority of the alert. E.g., high, medium, low.</td>
</tr>
<tr>
<td>summaryMessage</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>recommendationMessage</td>
<td>string</td>
<td>read-write</td>
<td></td>
</tr>
<tr>
<td>acknowledged (false)</td>
<td>boolean</td>
<td>read-write</td>
<td><strong>documentation</strong>: Defined as true if the user has acknowledge the alert and false otherwise.</td>
</tr>
<tr>
<td>errorID</td>
<td>string</td>
<td>read-write</td>
<td><strong>documentation</strong>: Internal ID representing the type of error of</td>
</tr>
</tbody>
</table>
Attributes inherited from TranswayObject:
  name description

Roles:

<table>
<thead>
<tr>
<th>Name</th>
<th>With</th>
<th>Association</th>
<th>Mult.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertType</td>
<td>AlertType</td>
<td>alert.alertType</td>
<td>1...1</td>
<td>Association</td>
</tr>
</tbody>
</table>

**Enumeration : eAlertPriority (alerts.Alert)**

Allowed Values:

LOW_AP MEDIUM_AP HIGH_AP
Keyword Index

A

abstract notion 39
adaptive 1, 2, 9, 11, 35, 37, 40, 72
  adaptive planning, 40, 41
add-on-armor 13, 24
Agent Server 97, 98
agents 1, 2, 3, 12, 15, 16, 26, 34-38, 40, 42, 45-48, 55, 58, 59, 61-63, 67-85, 88, 94, 98
  alerts 59, 75, 79, 80, 87
  Alerts Domain 58, 59
  collaboration 1, 37, 38
  Feasibility Agent 63
  icons 17
  Impediment Agent 27
  Planning Agent 63
  Truth Maintenance Agent 63
Agents Domain 88, 94
air channel 20
aircraft 9, 10, 36, 83
airfield 10
AJAX 96
Al Udeid 19, 24, 25, 31, 32
Alert Injector 80
alerts 59, 75, 79, 80, 87
Alerts Domain 58, 59
analysis pattern 40
Animation Domain 92
AOR (Area of Responsibility) 10, 11
Apache Tomcat 97, 98
API (Application Programming Interface) 2, 43, 62, 63, 89
APOD (Aerial Port of Debarkation) 10, 11, 17, 32
APOE (Aerial Port of Embarkation) 84
Apply Move 116, 118, 119, 127, 128
Apply Operators 118
ARC 15
Area of Interest 91
Area to Ignore 91
assessment 11, 35
associations 3, 12, 51, 52, 57, 61-63, 71
  self-association 3
attributes 51
authentication 64, 99
B

Backbase 96
backward chaining 72
Baker 84
barge 36
Barnes 67, 82, 85
Bartak 82, 83
Battiti 68, 69
Beamon 13
behavior 44
BEL (Business Engine Layer) 47
Best Solution 116, 119, 128
Blum 67, 82
BMNG (Blue Marble Next Generation) 65
Bojduj 129
Bonne 91
border 38, 40
bottleneck 85, 122, 128
Bowman 13
Bräysy 82, 83
bridges 91
BRL (Business Rule Layer) 47
Brooks 13
business rules 15, 46, 77

C

C++ 89
C130 aircraft 22
C17 aircraft 22
cache 52, 91, 122, 128
CADRC (Collaborative Agent Design Research Center) 33, 41, 42
CADRG (Compressed ARC Digitized Raster Graphics) 15
Cal Poly (California Polytechnic State University, San Luis Obispo) 33
candidate list 70
candidate move 78
cargo 9, 16, 55, 73, 76, 77, 79, 80, 84, 85
cargo loading 77
cargo unloading 77
Cassini 91
CDE (Corporate Data Environment) 2
CDM Technologies Inc. 3, 33, 37, 41, 42, 44
Chandrasekaran 43
Chappell 34
Charleston 19, 31
Charts Domain 93
check point 40
CIB (Controlled Image Base) 15, 65
CICE (Corporate Information-Centric Environment) 2
cities 91
classes 12, 51, 52, 62
CmapTools 113
COA (Course of Action) 9, 10
coalition 3
COI (Community of Interest) 51
collaboration 2, 11, 41-44, 46
collaborative tools 9
Colletti 67, 82
column generation technique 85
Combatant Command 1
combinatorial 71, 113
Combinatorial Optimization 113
command and control 37, 38
commercial 13
concurrency control 63
constraints 55, 69, 71, 72, 76, 78, 80, 83, 113, 115, 116, 121, 129
Constraint Checker 80
Constraint Logic Programming 72
constraint verification 129
Constraints Domain 55
constructive search 69
container 36
collection 3, 9, 12, 15, 17, 33, 35-37, 41, 43, 44, 47, 51
Controllers 90
conveyance 16, 18, 21, 28, 30, 36, 55, 73, 75-77, 79-81, 88, 93, 94, 116, 128
conveyance service 76
convoy 23
cookies 100
CORBA (Common Object Request Broker Architecture) 46
cost 41, 84, 127
Create Moves 116, 117, 119, 120, 124, 128, 129
Crino 13, 72, 73, 82
Current Solution 116-121, 125, 127-129
Cylindrical Equal-Areas 91

D

dams 91
DAO (Data Accessor Object) 61, 62
data 35-37, 44, 61, 78, 90	encyopedic data 53
data dictionary 131-156
wrapped data 44
data blackboard 48
data-centric 2, 37
data dictionary 131-156
data-processing bottleneck 34
Data Server 97, 98
Data Translator 80
Data Warehouse 38
DDL (Data Description Language) 51, 52
DDOC (Deployment and Distribution Operations Center) 1, 9, 11
decision-making 3, 11, 35
decision-support 1, 2, 9, 10, 12, 15, 33, 36-38, 40, 41, 43, 45, 46
decoupled 33, 35, 41, 42
delegate 61
delivery window 24
demonstration 17, 18, 20-32
deployment 3, 9, 11, 97
design pattern 44, 61
discovery 34, 35, 42, 46, 69
distributed 41, 42, 43, 47, 70, 130
distributed architecture 41
distributed-object server 38
distribution 3, 9, 12, 82, 83
diversification 67-70, 82
DoD (US Department of Defense) 11
Domain 89
Domain Loader 89, 91, 93, 94
Domain Model 47, 51, 52, 61-63, 71, 73, 80 81, 88-90, 96
Domain Views 90
Domain XML 89, 91-94
double-booked 80
Dover 31
driving hours 83
dynamic planning 75, 78
dynamic programming 76

E
eager fetching 63
Earthsat Natural Vue 65
editing 15, 16
effectiveness 10
efficiency 9, 10, 52
elevation 91
e-mail 44
encryption 99
encyclopedic data 53
Entities 89
Environment Domain 56, 57
Erl 33
ETD (Earliest Time of Delivery) 77, 79
evaluation function 77
events 3
Exhaustive Search 76, 78, 82
exponential time 128
extensibility 45, 65
extensible 45

F

facade 3, 44-47, 61, 88, 91, 96
Facade Java Bean 62
fault tolerance 63
Feasibility Agent 74, 79, 80
feedback 69, 79
Finding-Shortest-Path problem 116
firewall 98, 99
fixed wing aircraft 21
flexibility 34, 40, 41, 45
flow network 12, 85
flow planning 9
Floyd-Warshall algorithm 76
FWTools 66

G

GDAL (Geospatial Data Abstraction Library) 65
Gendreau 82, 83
genetic algorithms 12, 72, 82
geo-referenced 15, 17, 48, 57
GEOTIFF (Geostationary Earth Orbit Tagged Image File Format) 65
GIS (Geographic Information System) 66
Glasgow 27
Glover 48, 67-70, 114
Gnomonic 91
Graphical 92
Graphical Entities 89, 91, 93
group theory 82
GSG (Generic Space Generator) 49, 88, 90, 91, 93, 94
    GSG Mapping 88, 90, 91
    GSG Viewer 89

161
GSGActions 90, 92, 93
GTN (Global Transportation Network) 1
GUI (Graphical User Interface) 15, 88-90
GUlIL (Graphical User Interface Layer) 47

H

Harder 70, 71, 73, 119
hash code 125
hazardous material 40
hierarchical tree layout 90
Heuristic Search 76, 78
Hibernate 62, 63
home base 77
HTML (Hypertext Markup Language) 96
HTTP (Hyper-Text Transfer Protocol) 34, 97, 98
heuristics 67, 78
human-computer collaboration 79
hybrid heuristic 78

I

ICDM (Integrated Cooperative Decision Making) 3, 33, 37, 41-47, 49, 51, 52, 61, 62, 73, 87, 88, 90, 95
ICODES (Integrated Computerized Deployment System) 1, 17
IHMC (Institute for Human and Machine Cognition) 113
image pyramid 65
Imagery Service 61, 64, 65, 88, 90, 91, 96, 98
impediment 16, 17, 26, 27, 71, 77, 78, 80, 81, 91
Impediment Agent 73, 74, 81
Impediment Service 81
inference engine 38
information 2, 9, 34, 35, 38, 43, 44, 46, 47, 51, 90
information assurance 99
information tier 38, 46, 47, 48
information-centric 3, 37, 43, 44
inheritance 12, 40, 52, 62, 63
initialization 15, 18, 53, 63
integrity checking 99
intelligence 40
intensification 67-70, 82
intent 9, 35
interactions 90
interface 33
Internet 11, 35, 64, 99
interoperability 36, 37, 41
interoperability bridge 2, 3, 37
interpretation 44
in-transit visibility 2, 9, 11
IP address 99
Items Domain 55
iterative development 3

J

Java 12, 51, 61, 63, 87, 89, 96, 99, 119
Java Bean 61, 89
Java Binding 89
JavaScript 96
JDBC (Java database Connectivity) 98, 99
JDDOC (Joint Deployment and Distribution Operations Center) 1, 9, 11
JFC (Joint Forces Commander) 9, 10, 11
JFCT® (Joint Forces Collaborative Toolkit) 1, 17
JMS (Java Messaging Service) 75, 98, 99
JNDI (Java Naming and Directory Interface) 99
JOA (Joint Operating Area) 1, 9, 10
JOGL (Java Open Graphics Library) 89
joint 3
JOPES (Joint Operation Planning and Execution System) 1
JPEG2000 65
JSF (Java Server Faces) 96
jump search 71

K

Kaminski 13
Karaboga 48
KMES® (Knowledge Management Enterprise Services) 2, 3, 33, 35-40, 43, 47, 61
knowledge 34, 43, 51, 71, 84
knowledge management 2

L

Laguna 48, 67, 68, 70
LaPorte 69, 82
latitude 17, 65
lazy loading 63
LDAP (Lightweight Directory Access Protocol) 98
learning 35
lessons learned 2
lift asset 9
linear programming 84
load balancing 3, 42
load-planning 11, 36
Local Agents 89
local optima 67, 68, 82
lock contention 63
logic 2, 38, 40, 46, 47
logic tier 39, 46-48
Logical Entities 89, 93
Logical Facades 92
Lomow 41
longitude 17, 65
long-term memory 68-70, 129
LTD (Latest Time of Delivery) 77, 79

M

Map Domain 88, 91
Map Server 65, 66, 97, 98
Map Viewer 96
marshalling 10, 36
MASS (83, 84
McKinzie 83, 85
mediation 11
memory structures 68, 70, 114
Mercator 91
merit function 12
meta-characteristics 39, 40
metadata 52
metaheuristic 67, 82
military 13, 38, 81-85
minimum spanning tree problem 69
mission 9, 73, 76-78, 80, 81
Mission Generator 76
Mission Scheduler 76
mixed-integer 84, 85
MOG (Maximum on Ground) 77, 80
monitoring 35, 37, 40, 71, 78
Morton 84
Move 116, 117
move 68, 70, 73, 76-78, 116-118, 120, 121, 125, 126, 128
Move Generator 116, 117
Move Manager 70
move neighborhood 69, 70, 82
movement 9, 39
Mowbray 46
Mues 85
multi-layered 41, 45, 46
multi-modal 11
Multi-Processor Scheduling 113
multi-tier architecture 38
multi-tiered 41, 45, 46
MVC (Model-View-Controller) 88
Myers 44

N

Nanry 67, 82
.Net 34
net-centric 36, 37
net-centric architecture 36
Newcomer 41
NIPRNET (Non-Secure Internet Protocol Router Network) 99
NITF (National Imagery Transmission Format) 65
notion 3, 39
NP-Completeness 113
NP-hard 71, 78
NRMO (84
NSN (National Stock Number) 53

O

OAL (Object Access Layer) 46
OASIS (Organization for the Advancement of Structured Data) 33
objects 3, 12, 17, 38, 43-47, 51-53, 55, 57, 59, 62, 73, 80, 82, 87, 89, 91, 95, 96, 116
object function 70
objective function 13, 70, 84, 129
objective value 117, 119, 126, 127, 128
OML (Object Management Layer) 46, 47
ontology 1, 2, 3, 12, 15, 17, 33, 38, 39, 43-47, 51, 73, 76, 131-156
open architecture 11, 41
OpenGL 89
OpenTS Framework 70, 119
operating system 99, 100
operational 9, 11-15
Operator 116, 125, 127
opportunistic 26, 34, 47
opportunity cost 9, 10
optimistic locking 63
optimization 12-14, 26, 63, 72, 82-85, 114, 115, 127, 129
optimizing simulator 84
option bar 17
Oracle 62, 63, 98-100
organization 71
Organization Domain 57
ORM (Object relational Mapping) 38, 63
OTL (Object Transport Layer) 46

P

classification 36
pallet 36, 75-77
passenger 9, 13, 76, 77, 80
password 99
password authentication 99-101
password complexity 99-101
PC (Personal Computer) 100
penalty 83, 84, 123
performance 65, 128
persistence 38, 62
Persistence/Query Service 47, 52, 61, 62, 73, 88, 96
personnel 9, 13, 76, 77, 80
perspective 45, 46
pessimistic locking 63
Pham 48
Pickl 85
PKI-functionality 99
planning 1, 9, 11, 15, 35-37, 48, 55, 63, 67, 74, 78, 79, 113
Planning Agent 72-79, 81, 113-130
Planning Domain 59
plans 16
- creation 16
plug-and-play 41
PNG (Portable Network Graphics) 66
POD (Port of Debarkation) 15
POE (Port of Embarkation) 15
Pohl J. 34, 37, 38, 41, 44
Pohl K. 44, 45
polygon 16
polymorphism 63
polynomial time 115
port 10
portable 61
power transmission lines 91
pre-rendered 65
presentation 2, 38, 46, 47
presentation tier 39, 47, 48, 61
priority 9-12, 16, 24
process flow 79
pseudo code 119-127
Pyramid Builder 65

Q

query 53, 61-63, 65, 73, 88, 98
query metadata 91

R

rail 10, 36
Randall 69
random 70, 82
randomization 70
range 80
range-payload 84
raw trip 121-123, 128
RDBMS (Relational Database Management System) 38
readable 61
reasoning 3, 9, 15, 35, 37, 40, 43, 44, 46, 47
re-cycling 68, 70, 82
refueling 77, 82
registry 34, 42, 62, 63
Rego 67, 82, 83
relationship 12, 44, 47, 51
reliability 41, 65
re-planning 1, 2, 9-12, 15, 27, 36, 37, 48, 55, 62, 63, 67, 74, 75, 79, 113
reports 15, 16, 20, 31, 36, 48, 74, 87, 88, 92, 95, 96
alerts 48
bar charts 32, 88, 93, 94
charts 88, 93, 94
GANT charts 92, 94
graphical 31, 48, 87
representation 44
requests 58, 75, 96, 116, 121, 122, 128
requirements 16, 25, 26, 58, 71, 79, 80, 83, 113
Requirements Domain 58
REST 64, 65
rest periods 83
retrieval 35
re-usable 2, 33, 36, 45, 46
reward 84
risk 2, 9, 10, 36, 40
rivers 91
RMI (Remote Method Invocation) 75
road 10
Robinson 91
role-based authorization 99
Roli 67, 82
Roucairol 67, 82, 83
route 9, 16, 17, 25, 28, 36, 39, 62, 71, 74-76, 78, 81, 84, 90, 91, 94, 113, 116, 122, 127, 128
  leg 15
  segment 17, 76, 91, 92
  traversing 77
route leg 15
route segment 76
route traversing 77
Routing Service 76
Roy-Floyd algorithm 76
rule-based 3, 12, 13, 47, 84

S

San Luis Obispo 33
scalable 61, 65
scaling 42
Scenario Server 47, 61, 63, 64, 98
Scheduled Trip 116
scheduling 36, 71
Schrady 13
sea route 20
search 35, 38
search methodology 72, 73
Search Service 76, 78
search space 67-69, 72, 78, 82, 83, 117, 129
SEAWAY 1
security 13, 65, 99-101
Select Move 116-119, 125, 128
self-contained 3
self-diagnosis 3
self-healing 3
self-tuning 69
Semet 13
server 97, 99
service-oriented 2, 41, 47, 99
service-oriented architecture 47
ship 9, 10, 21, 23, 36
shortfall 16, 77, 127, 128
short-term memory 68-70, 129
shut down 15
Simulated Annealing 72, 82
simulation 13-15, 83, 84
SIPRNET (Secret Internet Protocol Router Network) 99
SOA (Service-Oriented Architecture) 2, 11, 33, 35, 41, 42, 47
SOAP (Service-Oriented Architecture Protocol) 34
software engineering 3
solution state 68
solution-based search 69
speed limit 83
spiral development 3
SPOD (Surface Port of Debarkation) 10, 11
SQL (Structured Query Language) 63
SSA (Supply Support Activity) 17
SSDM (Stochastic Sealift Deployment Model) 84
SSL (Secure Sockets Layer) 99-101
start-up 15
Stenbit 2
stochastic programming 84
storage 35, 36, 44
stove-piped architecture 36
strategic 9-14
strategic oscillation 68-70
struts 96
subscription 3, 35, 38, 43, 75, 80, 88
 priority 3
 profile 3, 43, 44
 query 3
subscription profile 43, 44
Subscription Service 15, 47, 61-63, 73, 88
Subscription Service Broker 63
supplies 10, 13, 16, 18, 28, 61, 71, 73, 62, 88, 93, 128
supply center 51, 61, 71, 73, 75, 79, 88, 91, 92
supply chain 12, 13
supportable 9
sustainment 2, 9-12
Sweet 89
synchronization 71

T

taboo 68, 70
Tabu List 70, 114, 116-118, 125, 128, 129
 reactive 68, 83
Tabu Search Framework 114, 115, 119, 128, 129
tactical 13, 38
Taillard 13
Talbi 71
Taylor 69
Tecchiolli 68
tenure 68, 69
terminology 3
Thai 34
theater 13, 38, 82, 85
Thick Client 2, 48, 49, 61-63, 87-94
Thin Client 2, 48, 61, 62, 87, 95, 96
threading 71
threat 16, 76, 127
three-dimensional 89
throughput 13
tiles 65, 66, 91
Time Domain 53
time window 81-83, 85, 118, 125, 127
TIRAC® 37
TL (Transport Layer) 47
token 100
Tomcat 99-101
tonnage 9, 10
transaction management 38
transfer 71, 73, 75, 80, 81, 92
Transfer Domain 92
Transfer Model Domain 88, 92
transshipment 77, 83-85
Traveling Salesman problem 71
traversal 80
traverse 127, 128
trip 71-73, 116, 117, 121-123, 125, 127-129
Trip Generator 116, 117, 128
trucks 18, 21, 23, 36
Truth Maintenance Agent 73, 74, 80, 81
two-dimensional 89
Types Domain 53, 54

U

UDDI (Universal Description, Discovery and Integration) 34
UML (Unified Modeling Language) 12, 51, 52, 62
uncertainty 84
Update Solution 116, 119, 128
URI (Uniform Resource Identifier) 34
URL (Uniform Resource Locator) 34, 65
use-case 63
user-interface 88
USTRANSCOM (United States Transportation Command) 1, 9, 63

V

vector 91
video card 89
Viewer 89-94
Visibility Agent 73
VPF (Vector Product Format) 15

W

W3C (World Wide Web Consortium) 34
W3C Web Architecture 65
warehouse 36
way point 91, 92
weather 26, 27, 36, 76, 81, 91
web 2, 11, 33, 34, 48, 65, 99
Web Client 95, 96, 99, 100
Web Server 97, 98
Web Services 99
Weber 12, 67, 71-73, 82-85, 129
widget 92, 94
Widget Manager 89, 90
Widgets 90
Wright 89
Wu 13, 83, 84

X-Y-Z

XMI (XML Metadata Interchange) 62
XML (Extensible Markup Language) 33, 34, 51, 52, 92-94

Yang 82, 83

Zahavi 46
zoom 65