

Toward a Meta-Level Framework for Agent-Supported Interoperation of Defense Simulations

Levent Yilmaz

Swetha Paspuleti

M&SNet: AMSL (The Auburn Modeling and Simulation Laboratory)
Computer Science and Software Engineering
Auburn University
Auburn, AL 36849
yilmaz@auburn.edu
paspusw@auburn.edu

Interoperation and dynamic composability of disparate simulations is a long-standing challenge within the defense modeling and simulation community. While both issues are extensively studied, a unified and coherent strategy that facilitates achieving seamless and transparent congruity among conceptual and realization spaces of simulations is still an elusive goal. In this paper, an agent-supported meta-level interoperation architecture is proposed to address these challenges. The strategy involves the introduction of an agent framework independent of the simulation infrastructure for explicit separation of interoperation protocols from the simulation environment. To this end, agent-based mediation, brokering, matchmaking, and facilitation services are suggested as critical components for interoperation and composability. The proposed approach is based on the following premises: 1) deployment of brokering protocols improves transparency and balances the workload in interoperation, 2) partial matchmaking mechanisms provide more efficient and effective model and data qualification strategies than the conventional keyword-based matching mechanism, and 3) automated mediation facilities that are deployed independently from the simulation infrastructure improve transparency and run-time extensibility. Stand-alone prototypes of the matchmaker and broker agents are developed to facilitate testing these propositions. We conclude by discussing the extension of the applicability of the proposed strategy to DoDAF architectural views and associated products.

Keywords: Interoperability, composability, intelligent agents, mediation, meta-simulation

1. Introduction

While dynamic composability, interoperation, and run-time extensibility in simulation modeling are highly sought [1], contemporary solutions often lack mechanisms for (dynamic) selection and assembly of models, as well as meaningful run-time interoperation among simulations. In particular, they are limited in dealing with 1) dynamically evolving content (i.e., data, model) needs of existing federated modeling and simulation (M&S) applications and 2) run-time inclusion of new simulations into a federated

system with their own encoding standards and behavioral constraints. Besides, existing interoperation strategies are not transparent to the actual simulation infrastructure. This makes reuse of existing simulations difficult. That is, interoperation of legacy simulation and components is often engineered via ad hoc extensions to existing simulations. As simulations join to and detach from the society of simulations, ad hoc strategies are likely to experience scalability problems.

To this end, the M&S community is taking steps to facilitate the improvement of integratability (Page [2]), interoperability (Hamilton and Yilmaz [3], Petty [4]), and composability (Harkrider and Lunceford [5], Petty and Weisel [6], Yilmaz [7]) of simulation models. For instance, Tolk [8] suggests the use of

open standards along with explicit delineation of model interdependencies as a prerequisite for a practical solution to composability. Web services (Lopes and Hamoudi [9]) and the management of their composition (Tosic et al. [10]) via rich metadata are suggested as an enabling technology to improve interoperability and composability. Within the interoperability domain, various models of interoperability (Morris et al. [11]) are suggested to facilitate the improvement of different dimensions (i.e., programmatic, constructive, and operational) of interoperability. Concomitantly, the Extensible Modeling and Simulation Framework (XMSF) aims to improve composability and dynamic extensibility of simulations via web-based open standards (Tolk [8]).

The work described in this paper involves the introduction of the notion of an agent-based meta-level framework for interoperation over the simulation infrastructure via explicit separation of run-time interoperation and composition mechanisms from the simulation environment. We do not claim to solve the interoperability and composability challenges (Davis and Anderson [1]) but rather suggest a novel strategy and framework within which such challenges can be addressed in a tractable and effective manner. In particular, agent-based mediation, brokering, matchmaking, and facilitation services are suggested as critical components for seamless and transparent interoperation and composition. The proposed approach is based on the following premises: 1) deployment of various forms of brokering among content producers and consumers within a federated simulation system improves transparency and balance of workload in interoperation; 2) concept-based partial matchmaking mechanisms provide more flexible (ability to adapt to changing schemas), efficient, and effective retrieval than conventional keyword-based discovery and matching mechanisms; 3) automated mediation facilities at higher levels of abstraction, and decoupled from the simulation infrastructure, improve transparency and run-time extensibility. Note that run-time extensibility and dynamic composability (Davis and Anderson [1]) imply situations where data, scenario, and service needs of potential simulations, which may join to and detach from the federation, may not be foreseen in advance at design time. This requires mechanisms that facilitate run-time recommendations for conceptual alignment of used models, along with mediation facilities that support seamless exchange of data and services within the realization space. The premise of the proposed strategy is based on the observation that for transparency and improvement of the reuse of simulations such interoperation protocols should be decoupled from the simulation infrastructure (Yilmaz [7]). To demonstrate feasibility of the proposed

approach, stand-alone prototypes of mediator, broker, and matchmaker components are developed as a testbed.

The rest of the paper is organized as follows. In section 2, we overview the related work in interoperability and delineate the challenges addressed in this work. Section 3 presents the components of the meta-level agent framework that is proposed for orchestration of a society of simulations for interoperation. The stand-alone prototypes of the broker and matchmaker agents are discussed in section 4. Finally, in section 5, we conclude by discussing potential issues in developing next generation interoperation protocols in terms of the components of the proposed meta-level framework.

2. Issues in Interoperability and Composability

Supporting cooperative interaction among globally available simulations requires seamless exchange and sharing of data, information, knowledge, and models. The constant evolution and change of these resources complicate seamless discovery, location, and retrieval of “sufficiently” relevant services for the task at hand. Furthermore, the heterogeneity underlying the syntax and semantics of these services requires intelligent facilitators and mediators for content exchange among producers and consumers that have different world-views in defining and interpreting such services.

Emergent technologies related to building infrastructures for sharing and exchange of resources and services are promising for building the desired capabilities. The latest in this area is the grid services, an extension of web services. Web services have several advantages over other distributed system technologies like RMI and CORBA, as they are platform and language independent and use semantic web concepts to provide an open and extensible infrastructure. But it has disadvantages of overhead and lack of versatility in providing services such as persistency, notification, and lifecycle management. While grid services are highly promoted for resource sharing (Foster et al. [13, 14]), the existing infrastructure does not provide semantic matchmaking and various intelligent brokering mechanisms discussed in the proposed work. The dynamic matchmaking, semantic discovery of services (information, data, models etc.), and their brokering can be deployed over the facilities provided by basic grid services like registry, service discovery, subscription, and notification services.

2.1 The Limitations of Conventional Methods

Common problems with existing protocols for matchmaking and brokering are that they are rigid (not flexible to adapt to evolving content schemas) and assume the existence of a set of homogeneous resources that are pre-engineered for structural and semantic alignment. Yet, semantic, descriptive, structural, and heterogeneous conflicts (see section 3.2) are pervasive. Furthermore, partial matchmaking based on similarity metrics is not employed. Existing keyword-based techniques that are based, for instance, on XML tag comparisons are not powerful in taking into account the meaning and contextual constraints associated with a simulation model. Improved intelligent matchmaking techniques along with data and information mediation and facilitation (i.e., content-based routing) strategies are critical to facilitate (dynamic) selection and assembly of models, as well as their run-time interoperation. Furthermore, existing brokering techniques are not designed with the possibility of real-time decision making activities, where information needs to emerge on-the-fly.

In proposing the desired infrastructure, agents (Sycara et al. [15], Genesereth and Ketchpel [16], Hyacinth [17]), grid services (Foster et al. [14]), and brokering/matchmaking protocols are examined. Agents are intelligent computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems. Table 1 presents widely accepted and used types of software agents. The concept of interface/user agents, information agents, and middle agents are used in this work.

2.2 Basic Requirements

To achieve run-time extensibility and seamless interoperation among simulations, conceptual and realization spaces need to be aligned. As such, interaction must be meaningful between all composed services; in other words, each service has to know *what* data (model) is located *where*, the *meaning* of data (model) and its *context*, and into what *format* the data (model) have to be transformed to be used in respective services composed into a distributed application within the overall system. The following are the minimal requirements for an infrastructure that aims to support interoperation among simulations in a federated system (Tolk and Diallo [18]).

- *Administration* is the process of managing the information exchange needs that exist between the services. Administration involves the overall information management process for the service architecture. Location, discovery, and

Table 1. Types of agents

Agent Type	Description
Reactive agents	Simple agents that act based on stimulus/response based on observations in the current state of the environment
Mobile agents	Mobile agents that move from one system to the other in the network; they improve performance by moving the agent to the data rather than moving data to the agent
Collaborative agents	Autonomous agents that interact with other agents and share information for various purposes
Interface/User agents	These agents interact with the users to provide assistance in using a particular application.
Middle agents	Agents that help locate other agents are called middle agents (Sycara et al. [15]).
Information agents	Information agents are responsible for managing information obtained from distributed resources.

retrieval of content are critical components of administration.

- *Management* involves identifying, clarifying, defining, and standardizing the meaning of content.
- *Alignment* ensures that the data to be exchanged exist in the participating systems as an information entity or that the necessary information can be derived from the available models and services published by participants.
- *Transformation* is the technical process of aggregation and/or disaggregation of the information entities of the embedding systems to match the information exchange requirements.

3. The Strategy: Decoupling Interoperation via a Meta-Level Agent Framework

We propose a meta-level framework that constitutes various agents that coordinate and orchestrate seamless information, data, and service exchange among simulations. To support the requirements listed in section 2, an agent-based strategy is suggested. Figure 1 depicts an agent organization that constitutes facilitator, mediator, broker, and matchmaker agents that perform the necessary management, alignment, and transformation functions.

Furthermore, the agent organization aims to decouple the simulation from the intricate details of instantiation and interoperation of a family of models to avoid explicit assumptions and facilitate seamless reconfiguration with alternative ensembles. This way, the agent organization abstracts the simulation instantiation and interoperation process. It helps make a simulation system independent of how its models are created, composed, and represented. The organizational domain encapsulates the knowledge about which models the simulation uses. Furthermore, the concrete organization hides the details about how simulation programs for these models are created and composed together. Therefore, the decoupling of the instantiation and interoperation processes from the simulation infrastructure gives significant flexibility in terms of *what* gets instantiated and exchanged, *who* instantiates it, *how* it gets created and transformed, and *when*. Figure 1 illustrates how the interoperation framework and its components are positioned with respect to the infrastructure.

While the interoperation protocols and the overall meta-level are not yet realized, individual components of the framework are prototyped in a stand-alone manner to gain insight about their design, implementation, and potential roles within next generation intelligent agent-mediated interoperation protocols. The following paragraphs describe the facilitator, mediator, broker, and matchmaker agents, with particular focus on details of the brokering and matchmaking aspects of the framework.

3.1 The Facilitator Agent

The facilitator agent acts as a gateway between the simulation infrastructure and the agent organization that orchestrates the simulation interoperation. Simulations join a society of simulations by registering their facilitator with the meta-level interoperation protocol. As a controller, the facilitator agent is aware of the capabilities and needs of the simulation service with which it is associated. The requests coming from the simulation domain will be delegated to brokering, matchmaking, and mediation agents in accordance with the embedded interoperation protocol that facilitate seamless data discovery, location, retrieval, and transformation.

3.2 The Mediator Agent

The mediator agent is responsible for converting simulation content to/from a common reference model (i.e., C2IEDM). To facilitate mediation, conflicts between the assumptions and obligations of simulations need to be resolved. As discussed by Tolk and Diallo [18], four types of conflicts are common between the desired and provided services.

- *Semantic* conflicts occur when the local schemata objects must be aggregated or disaggregated but fail to match.
- *Descriptive* conflicts occur when the same concept is described in term of synonyms, homonyms, different attributes, and values.

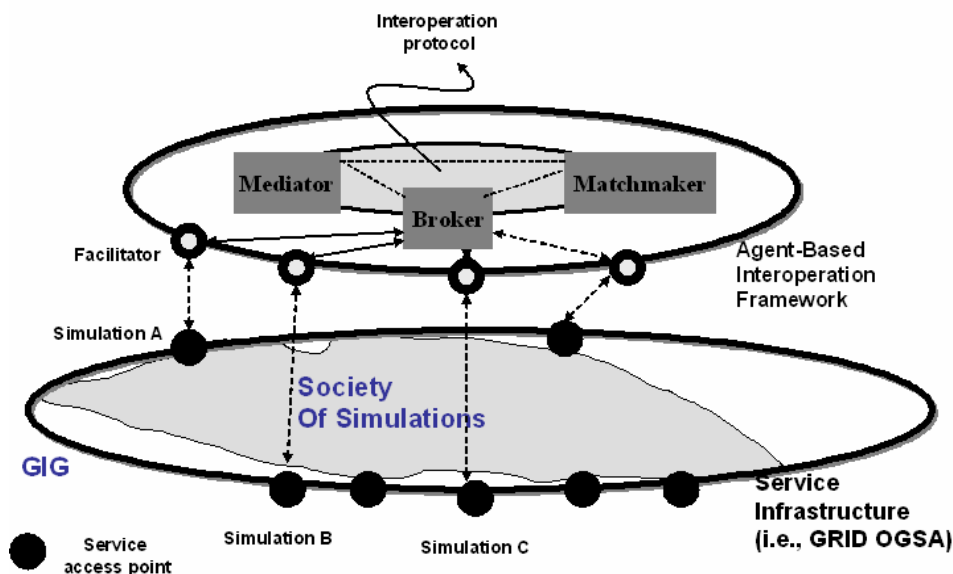


Figure 1. Agent-based meta-level interoperation: the organizational domain

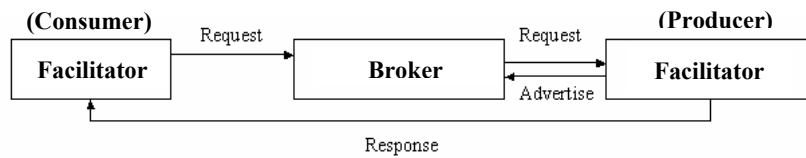


Figure 2a. The recruitment protocol

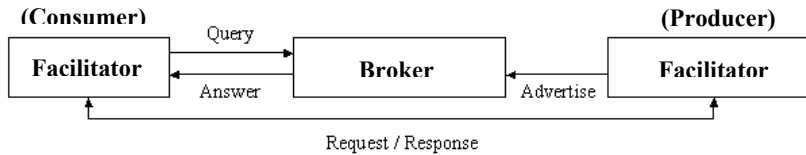


Figure 2b. The recommendation protocol

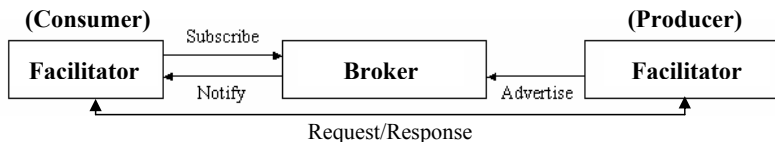


Figure 2c. The notification protocol

- *Heterogeneous* conflicts occur when concepts are described on terms of substantially different methodologies.
- *Structural* conflicts occur when concepts are defined in terms of different structures in the schemata (metadata).

The mediation agent uses a common reference model to convert between the data encoding standards. Note that the same strategy is applicable to models. That is, models can be converted from one formalism to another to facilitate the use of disparate simulations developed within distinct conceptual frameworks deployed in the infrastructure.

3.3 The Broker Agent

The interaction between content (i.e., data, model) requesting facilitators (consumers) and potential service providers (producers) are achieved via flexible mechanisms that can vary depending on the characteristics of the application domain. The brokering protocols that are implemented in our prototype are as follows:

- *Recruiting*: The consumer consults with the broker agent to select a producer that provides the most relevant content via matchmaking and delegates

the request to the facilitator of that producer (i.e., content management system, simulation application, or C2 system). However, any subsequent responses from the producer are sent directly to the consumer through their facilitators. The interactions are shown in Figure 2a.

- *Recommendation*: The consumer consults with the broker agent to perform matchmaking to retrieve a list of matched content specifications. The broker agent presents the list of specifications (metadata, metamodel) along with their rankings to the consumer, which then selects and contacts the desired service provider. The interactions are shown in Figure 2b.
- *Notification/Subscription*: The consumer subscribes via a broker to a specific type of content and is notified when it is available or published by a producer. The facilitator (consumer) then interacts directly with the facilitator (producer) to obtain the desired data/model. The interactions for this brokering mechanism are shown in Figure 2c.

Brokering among content producers and consumers in a federated simulation system brings flexibility via fine grain interoperation that takes specific constraints of individual simulations into account. For instance, two simulations that use a common ontology and vocabulary may not require mediation. In that case,

the recommendation protocol would be sufficient, as the producer and the consumer can directly exchange data and services through their facilitators. On the other hand, when conflicts among simulations need to be resolved, the recruitment protocol can be used so that the broker can submit the requests to the producer along with instructions for mediation. In cases where producer and consumer need to be synchronized, the broker agent can shift to a notification protocol between two specific facilitators.

3.4 The Matchmaker Agent

For dissemination and recommendation of services, various protocols in different contexts have been studied; see Durfee et al. [19], Kuokka and Harada [20], Kawamura et al. [21], and Sycara et al. [15]. Matchmaking is the cooperative partnership between information providers and consumers along with the help of an intelligent facilitator (Genesereth [22]). Using this approach information providers actively publish their capabilities and services to the matchmaker, and the consumers send requests for their desired information to the matchmaker.

Matchmaking enables the providers and requesters to exchange dynamically changing information in a more effective and active manner than traditional methods. Matchmaking mechanisms have been widely used in various applications and fields where information changes rapidly, such as product development and crisis management (Kuokka and Harada [20]). Yet, dynamic interoperation of such services is critical for coherent operation of the overall system. Furthermore, a matchmaking strategy with quantitative distance metrics provides insight about the extent of alignment needed for two (data) models to interoperate. Hence, a matchmaking agent needs to cooperate with the mediator agent within the envisioned interoperation protocol to improve the degree of automation in interoperation.

Qualifying content specification objects and then ranking them requires interpretation of specifications to compute relevance metrics. The matchmaker agent deployed in our prototype uses multiple filters at different levels of conceptual granularity. Concept-level matchmaking measures the distance between the requested and target objects to qualify concepts within a common domain ontology used by the mediation

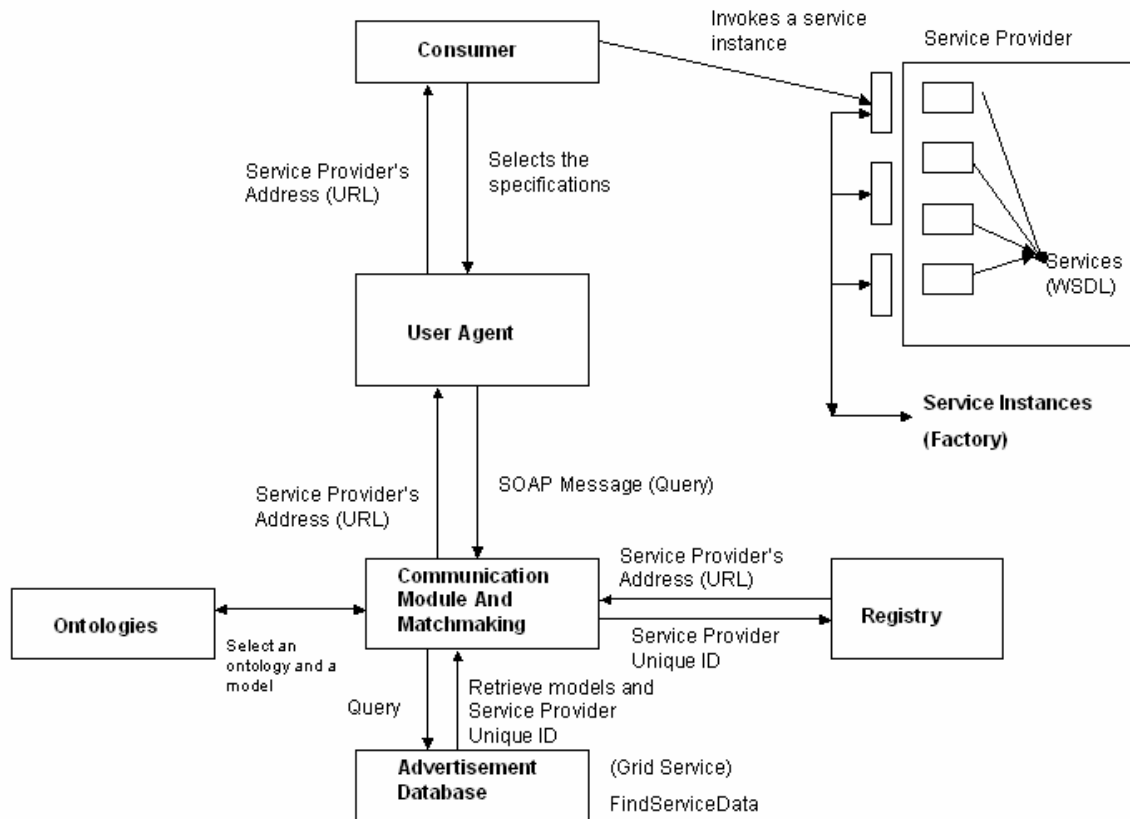


Figure 3. Components of the matchmaking agent

component. Attribute-level matchmaking focuses on measuring the effort involved in aligning the properties of the requested object specification and the qualified concepts. Finally, structure similarity analysis measures the similarity between the hierarchical structures of concepts that are found to be conceptually and property-wise similar.

3.4.1 A Strategy for the Design and Implementation of the Matchmaker Agent

The stand-alone implementations of the broker and matchmaker agents are realized over the Globus Toolkit 3 (GT3) using Tomcat Server 3.2.3, Simple Object Access Protocol (SOAP 2.2), Darpa Agent Markup Language (DAML), DAML Query Language (DQL), Java JDK, JRE 1.4.0, Jena Parser 1.6.1, and MySQL Server.

The components of the matchmaker agent are the communication module, matchmaking engine, ontologies, advertisement database, and the registry; see Figure 3. The communication module forms the entry point for the agent. All requests to the agent pass through the communication module to be processed by the matchmaking module that uses the domain ontology, advertisement database, and registry components. Tasks performed by a matchmaker agent are described as follows:

- The matchmaking agent performs *concept-level matchmaking* (see section 3.4.2), contacts the database server, and retrieves the model specifications that match the desired one.
- It then computes the *linear distance metric* and *structure distance metric* for each model qualified during the concept-level matchmaking phase.
- It also calls the client associated with service provider's grid service, to retrieve the model instances, and returns the model instances back to the brokering agent.
- A matchmaking agent is also responsible for receiving the model advertisements from the service provider agent and store.

The conceptual and structural matchmaking mechanisms via distance metrics constitute the foundation of the design of the matchmaking agents.

3.4.2 Concept-Level Matchmaking Strategy

The qualification of a content based on its conceptual definition involves the determination of the similarity of the requested concept to existing concepts defined within the domain ontology. Consider, for instance, the partial view of the military vehicle ontology shown in Figure 4. The ontology presents a partially specified conceptual domain ontology using the subsumption

relation between the concepts. Each model or capability is defined as a concept in the ontology, and each child node in the classification tree is connected to its parent through an *is-a* relation. For instance, in the example military vehicle ontology both the *M1Abrams* and *USTank* are of type *Tank*. Suppose that the experimental frame requires a vehicle of type *Tank* based on the presented military vehicle taxonomy. Using the ontology, a matchmaking engine needs to determine the relevance of the request to existing published models based on their conceptual similarity in the context of the used domain ontology. Based on the location of the requested concept (*Q*) required by the consumer and the published model concept (*A*), which is being examined for qualification, there are four cases to consider:

- 1) If $Q = A$ then the requested concept and published (advertised) capability are equivalent and there is an exact match between the concepts.
- 2) If *A* is a subtype of *Q* then the published concept is substitutable for the requested concept.
- 3) If *Q* is a subtype of *A* then the model can be used with slight modifications to perform the desired tasks.
- 4) If there is no subsumption relation between the published concept *A* and the requested concept *Q* then the specification matching fails.

Based on this contextual information, the degree of match can simply be defined as the ratio of the length of the minimum path between *Q* and *A* to the height of the taxonomy (classification) tree. The smaller the ratio, the more relevant is the model under examination.

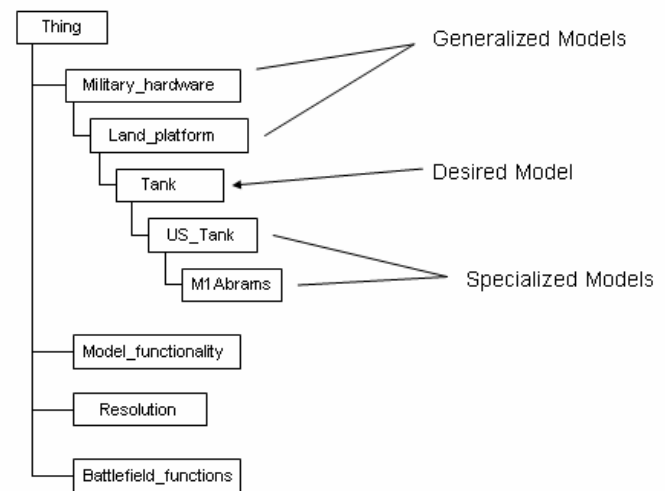


Figure 4. A partial view of the military models ontology

3.4.3 Attribute- and Structure-Based Matchmaking

The second stage of matchmaking involves attribute and content structure similarity analysis. Hierarchical structures of the requested and available content are compared, and the cost for alignment is computed. The degree of match determines the extent to which the requested content is structurally and content-wise similar to the provided entity. The second phase of filtering is divided into two levels: linear distance metric and structural distance metric. The linear distance metric calculates the effort to align the properties of compared entities, whereas the structure metric measures similarity of the hierarchical structure for complex entities.

The Linear Distance Metric (LDM): The algorithm presented by Yilmaz [23] is extended to realize the first stage of the matchmaking process. The cost of aligning the requested content hierarchy onto the hierarchy that is being qualified is determined. Sample hierarchies that depict concept structures are shown in Figure 5. The first step in comparing the specifications at that level is to serialize them via pre-order traversal. The result is a list of attributes. The distance metric measures the minimal cost of possible insertions, deletions, and moves to align the source content specification with the target specification (i.e., request).

The computed distance is then normalized into the range of (0, 1). The metric is formulated as follows:

$$LDM = \frac{Cost_i \times N_i + Cost_m \times N_m + Cost_d \times N_d}{\max(Cost_i, Cost_m, Cost_d) \times L_{stream}}$$

Insertion, deletion, or move of an attribute is defined in terms of $Cost_i$, $Cost_d$, and $Cost_m$. The numbers for each type of operations are defined as N_i , N_d , and N_m . Hence, $Cost_i \times N_i + Cost_m \times N_m + Cost_d \times N_d$ stands for the total cost of the transformation operations, whereas $\max(Cost_i, Cost_m, Cost_d) \times L_{stream}$ defines the possible maximum cost. Here, L_{stream} is the length of the source attribute stream being transformed (aligned) to the target attribute stream. We choose to align the qualified entity to the requested entity. In that case, the costs of the transformation operators can serve as tuning parameters for the metric. For instance, one can argue that the insertions are more costly, because they represent missing attributes. Conversely, practitioners can argue about the importance of deletions, because they may represent attributes that may unnecessarily complicate the object.

The range of values denoted by the LDM metric, for all practical intents and purposes, is between 0 and 1. However, technically, it is possible to obtain a value greater than 1.0 if, for instance, all attributes are deleted, and some new attributes are inserted. But such conditions are not likely. The distinction between strong and weak correspondence can be based on the statistical correlation rules of thumb. In particular, a measurement that is less than 0.2 can be considered

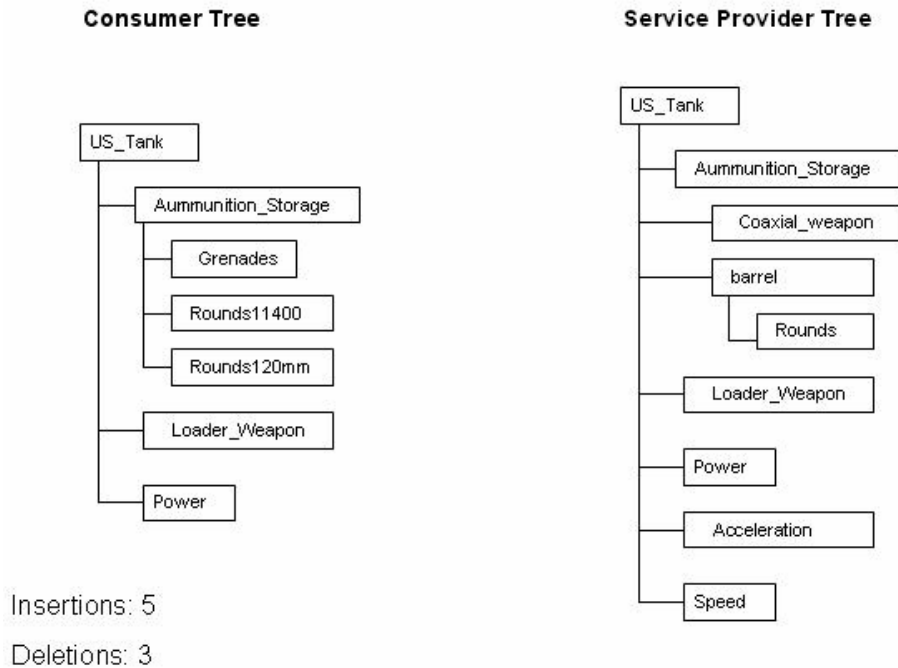


Figure 5. Hierarchical concept structures

as strong correspondence or alignment, whereas a value larger than 0.5 is a good indicator for weak correspondence. To compute the optimal operation sequence that transforms a source attribute stream

$$M = \langle (a_1, t_{11}), (a_2, t_{12}), \dots, (a_n, t_{1m}) \rangle$$

to a target stream

$$S = \langle (b_2, t_{21}), (b_2, t_{22}), \dots, (b_m, t_{2n}) \rangle,$$

the following dynamic programming algorithm works recursively and finds distances $d(a^i, b^j)$, where a^i and b^j indicate the initial segments of the attribute types $a_1 \dots a_i$ and $b_1 \dots b_j$ (including the case where the initial segment has length 0). Here, we assume that every move operation has a constant cost, C , for the sake of brevity of the discussion. Later, we will drop this cost, as substitutions can be defined in terms of insertions and deletions. The algorithm computes $d(a^i, b^j)$ for successively larger values of i and j , finally reaching $d(a^n, b^n)$, which is the distance between the two streams. The recursion starts with the base case $d(a^0, b^0) = 0$. The recursion equation for arbitrary subsequences is defined recursively as follows:

$$\begin{aligned} d(a^{i-1}, b^j) &= \text{Cost}(\text{delete}(a_i)) \\ d(a^i, b^j) &= \min [d(a^{i-1}, b^{j-1}) + \text{Cost}(\text{move}(a_i, b_j))] \\ d(a^i, b^{j-1}) &= \text{Cost}(\text{insert}(b_j)). \end{aligned}$$

Given the recurrence relation above, the algorithm is as follows.

ALGORITHM:

$$\begin{aligned} &d(a^0, b^0) = 0; \\ &\text{for } i = 1 \text{ to } m \text{ do } d(a^i, b^0) = \text{Cost}(\text{delete}(a_i)) \\ &\text{for } j = 1 \text{ to } n \text{ do } d(a^0, b^j) = \text{Cost}(\text{insert}(b_j)) \\ &\text{for } i = 1 \text{ to } m \\ &\quad \text{for } j = 1 \text{ to } n \text{ do} \\ &\quad \text{if } a_i = b_j \text{ then } \text{Cost}(\text{move}(a_i, b_j)) = C \\ &\quad \text{else} \\ &\quad \quad \text{Cost}(\text{move}(a_i, b_j)) = \text{Cost}(\text{delete}(a_i)) \\ &\quad \quad \quad + \text{Cost}(\text{insert}(b_j)) \\ &\quad \quad d(a^i, b^j) = \min\{ d(a^{i-1}, b^j) + \text{Cost}(\text{delete}(a_i)), \\ &\quad \quad \quad d(a^{i-1}, b^{j-1}) + \text{Cost}(\text{move}(a_i, b_j)), \\ &\quad \quad \quad d(a^i, b^{j-1}) + \text{Cost}(\text{insert}(b_j)) \} \\ &\text{return } d(a^m, b^n) \\ &\text{end.} \end{aligned}$$

It is straightforward to see that the algorithm realizes the recurrence relation defined above. The result of the algorithm is the optimal cost of transforming the source attribute stream into the target attribute stream.

Using the optimal transformation algorithm and applying the linear structure metric with $\text{Cost}_i = 2$, $\text{Cost}_d = 1$, and $\text{Cost}_m = 0$ (move operation is dropped, because it can be defined in terms of insertions and deletions), we obtain the following formula:

$$\begin{aligned} \text{LDM} &= \frac{\text{Cost}_i \times N_i + \text{Cost}_m \times N_m + \text{Cost}_d \times N_d}{\max(\text{Cost}_i, \text{Cost}_m, \text{Cost}_d) \times L_{\text{stream}}} \\ &= ((2 \times 3) + (1 \times 5)) / (2 \times 8) = 0.68. \end{aligned}$$

Note that, as shown in Figure 6, the number of insertions, N_i , is 3. This is due to the insertion of the attributes, "Grenades," "Rounds11400," and "Rounds120mm." The number of necessary deletions for alignment is defined by N_d , and it is 5.

Structural Distance Metric: The structural distance metric determines the degree to which two trees are structurally similar. A tree represents the hierarchical representation of a composite entity. For each pair of the trees, a structure metric is calculated to determine the degree of similarity between the structures. The method is an extension of the similarity algorithm presented by Romanowski and Nagi [24]. In our approach, the two trees are divided into groups of single-level subtrees and terminal subtrees. A single-level subtree is a parent node with its children. The children may or may not be leaf nodes. A terminal subtree is a parent node with its children, which are all leaf nodes. To illustrate the basic operation of the algorithm, consider the two trees, T_1 (consumer tree) and T_2 , as shown in Figure 7. First, an adjacency matrix is calculated for each of the trees. If the node in the column is a child of the node in the row, their corresponding entry will have a "1," or else "0." If the sizes of the two matrices differ, the smaller matrix is filled with rows and columns of 0's, so that the two matrices are of the same size. After the adjacency matrices of T_1 and T_2 are calculated, each tree is grouped into pairs of terminal subtrees and single subtrees. Next, each subtree in T_1 is compared with every subtree in T_2 . In trees that exactly match, the children nodes are removed from the subtrees. For instance, in Figure 7, the trees rooted at d in T_1 and T_2 match and the trees rooted at j in T_1 and T_2 match. The children $f, g, m,$ and n are removed from the trees. The trees shown in Figure 8 depict the configuration after removing the exact matches.

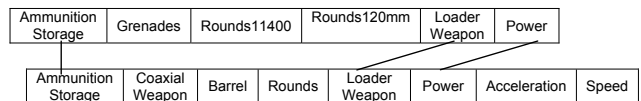


Figure 6. The insertions and deletion operations

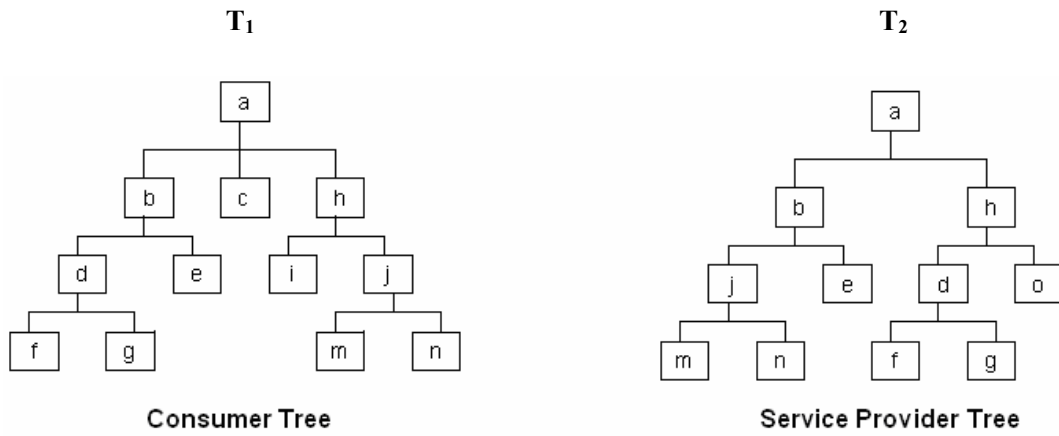


Figure 7. Trees and their corresponding adjacency matrices

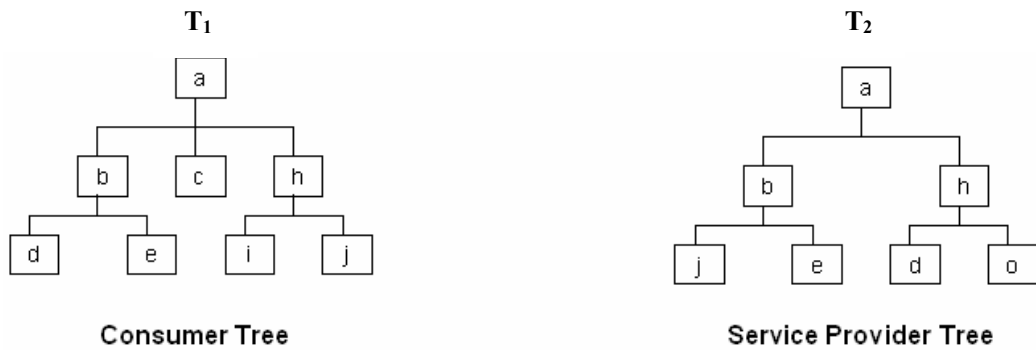


Figure 8. Reduced concept hierarchies

From the remaining subtrees, the distance between subtrees of T_1 with each subtree of T_2 is calculated. For each subtree of T_1 , the subtree of T_2 that has the minimum distance is obtained. The distance between subtrees rooted at b in both trees is 2, as the node e matches but the nodes d and j do not match. Similarly, the distance between the subtree rooted at b in T_1 and the subtree rooted at h in T_2 is also 2. Hence the minimum distance for the subtree rooted at b in T_1 is 2. The minimum distance for subtree rooted at h is 2. The distance for the tree rooted at a is 1, as b and h match but c does not. The structure metric for T_1 and T_2 is the sum of distances between all these pairs divided by n^2 , where n is the number of nodes in the larger tree. In the example, the distance is $(2 + 2 + 1)/n^2$, where n is 12, so structure metric = $(2 + 2 + 1)/144 = 0.035$.

The structure metric is calculated for each pair of requested and qualified concept hierarchies, and the one with the least value forms the best match. The algorithm is presented as follows:

ALGORITHM:

Input: Unordered trees $T_1, T_2, |T_1| \leq |T_2|$

Output: Structural distance metric determining the distance between two trees, T_1 and T_2 :

- 1) Form adjacency matrices of T_1 and T_2 , represented by A and B , respectively.
- 2) If $|B| > |A|$ then augment A with $|B| - |A|$ rows and columns of zeros at the right end and bottom of the matrix.
- 3) Calculate the initial structure metric $S_{ab} = D/n^2$ with $D = |A - B|$, where D represents all nodes that are in A and not in B and nodes that are in B but not in A . n is the number of nodes in the larger tree.
- 4) Find exactly matching subtrees
 For $i = 1$ to m , where m is the number of subtrees in B .
 Start at right side of matrix B and move to left.
 Find the first terminal subtree tb_i rooted at n .

Search A for an exactly matching terminal subtree t_1 with root n .

If t_1 and tb_i are exactly matching subtrees then reduce both A and B by removing the rows and columns corresponding to the child nodes of t_1 and tb_i .

Repeat for single-level subtrees in A and B .

If no more exact matches remain, continue.

- 5) Find paired subtrees using $\text{TreeMatch}(T_1, T_2)$. Get D from TreeMatch .
- 6) Calculate $S_{ab} = D/n^2$.
- 7) Return S_{ab} .

Function $\text{TreeMatch}(T_a, T_b)$

- 1) Get all subtrees of T_1 and T_2 with a minimum distance between them.
- 2) $D = \text{Sum of } \min(t_1, t_2) \text{ for all subtree pairs } \{t_1\}, \{t_2\}$.
- 3) Return D .

4. The Design and Implementation of the Broker and Matchmaker Agents

The mediator, broker, and matchmaker agents are developed as stand-alone components. In this section, we present a brief synopsis of the matchmaker and broker components developed at the Auburn Modeling and Simulation Laboratory (AMSL). The brokering protocols and the matchmaking techniques discussed in section 3 are implemented within the context of an Agent-Mediated Model Retrieval and Composition System (AMRCS), which was developed at the AMSL. Figure 9 illustrates one of the ontologies used in AMRCS, whereas Figure 10 shows the result of the first level of concept matchmaking with respect to a query that aims to locate entities that can be used in place of a *tank* entity.

The underlying infrastructure of the system is the grid, and the implementation is built over the Globus Toolkit 3 (GT3). The technologies used in building the prototype are Globus Toolkit, Tomcat Server 3.2.3,



Figure 9. The military vehicle ontology

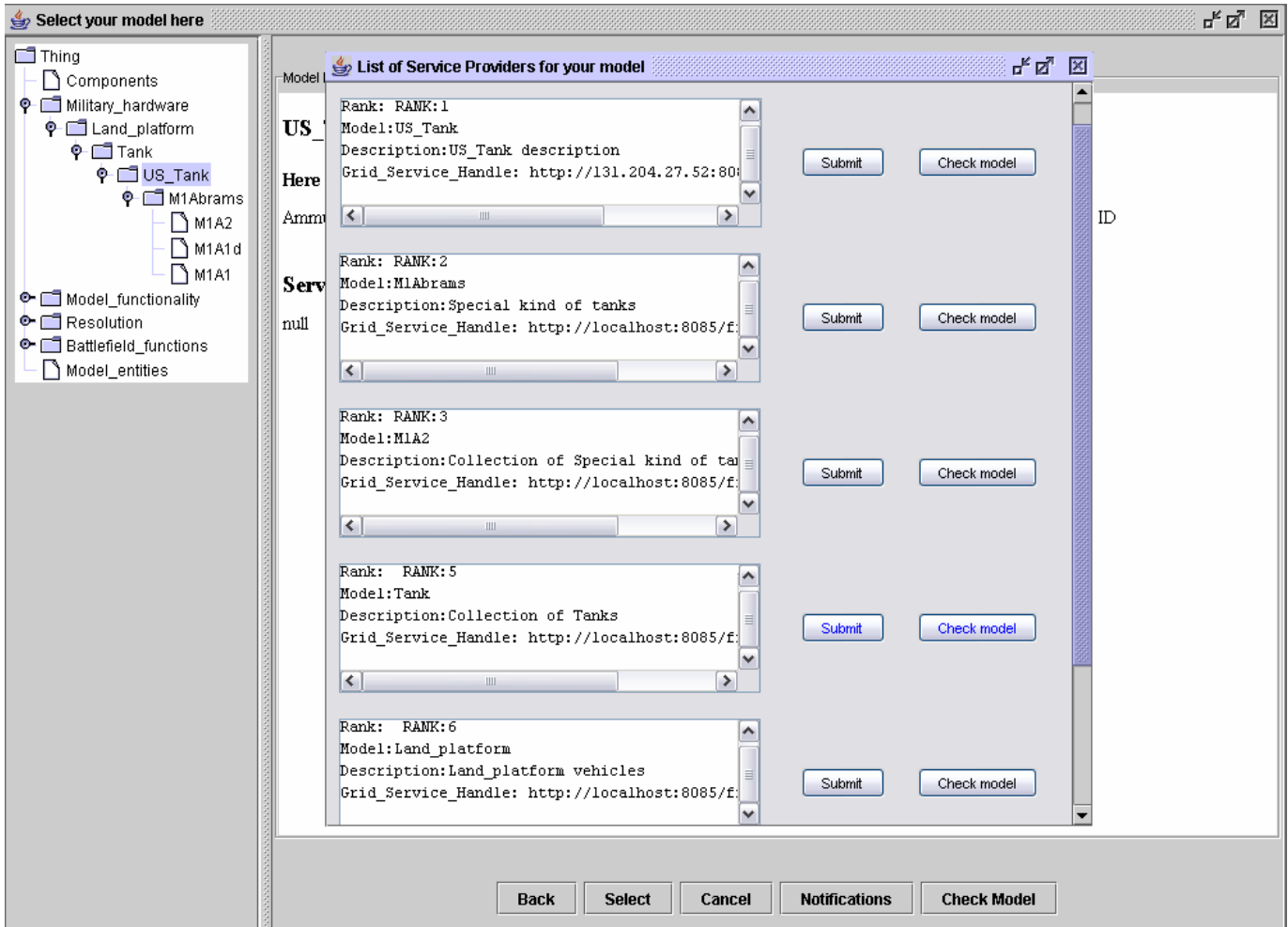


Figure 10. The results of conceptual matchmaking

SOAP 2.2, DAML, DQL, Java JDK, JRE 1.4.0, Jena Parser 1.6.1, and MySQL Server. The knowledge base used by the matchmaker is a collection of ontologies representing the domain of military models.

The military models are classified into several domains, and each domain is associated with an ontology. The DAML language was developed as an extension to XML, and the Resource Description Framework (RDF) is used to express the ontologies.

In our prototype, as a result of the first-level matchmaking for US Tank, the entities designated by "US_Tank," "M1Abrams," and "Tank" are qualified. The concept hierarchies associated with these models are shown in Figure 11. The attributes of the desired entity, selected by the facilitator of the consumer, are Ammunition_Storage, Rounds120mm, barrel, rounds, and Speed. The structure of the requested entity is shown in Figure 5.

It is clear, as depicted by Figure 11, that the degree of match against "M1Abrams" is stronger than the "US_Tank" entity, which then exhibits a stronger match compared to the "Tank." This is supported by the distance metrics presented in Figure 12. In our prototype, under the recommendation protocol, the placeholder component for the facilitator selects from the list of qualified models recommended by the broker, and the brokering agent calls the service provider to query its model instances database. The facilitator for the service provider then provides to the broker agent the values of the attributes for the selected entity. The brokering agent forms a DQL query in terms of the attributes, in conjunction with the input property values, and sends it to the grid service to retrieve model instances.

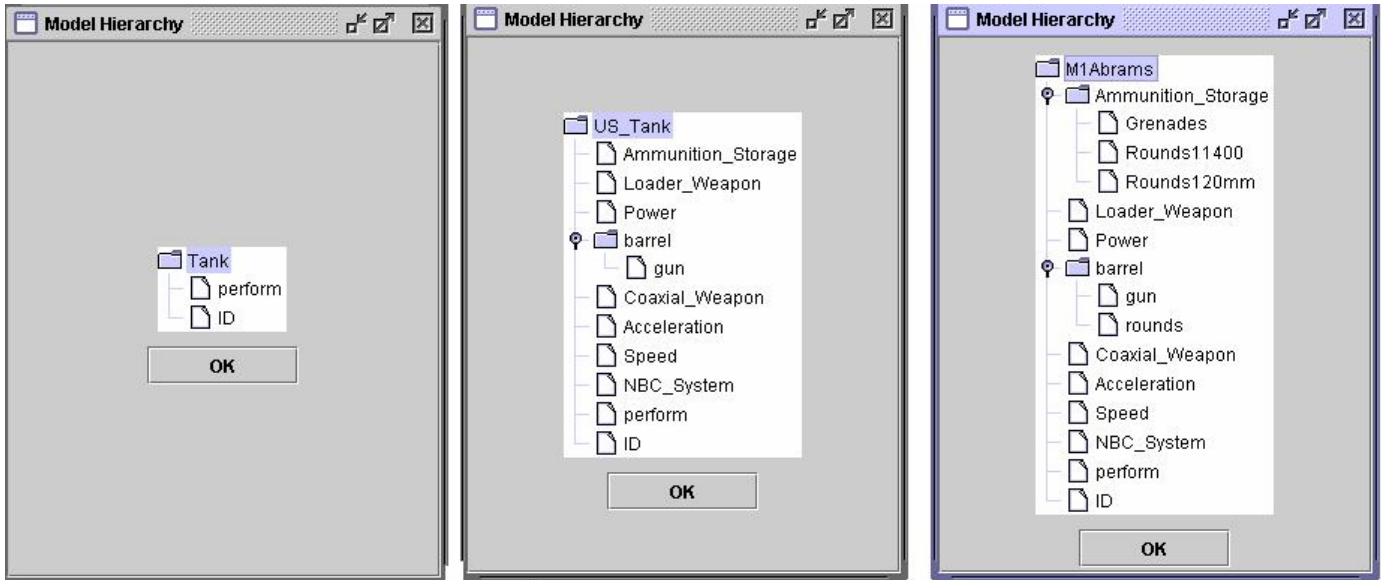


Figure 11. Concept structures of Tank, US_Tank and M1Abrams

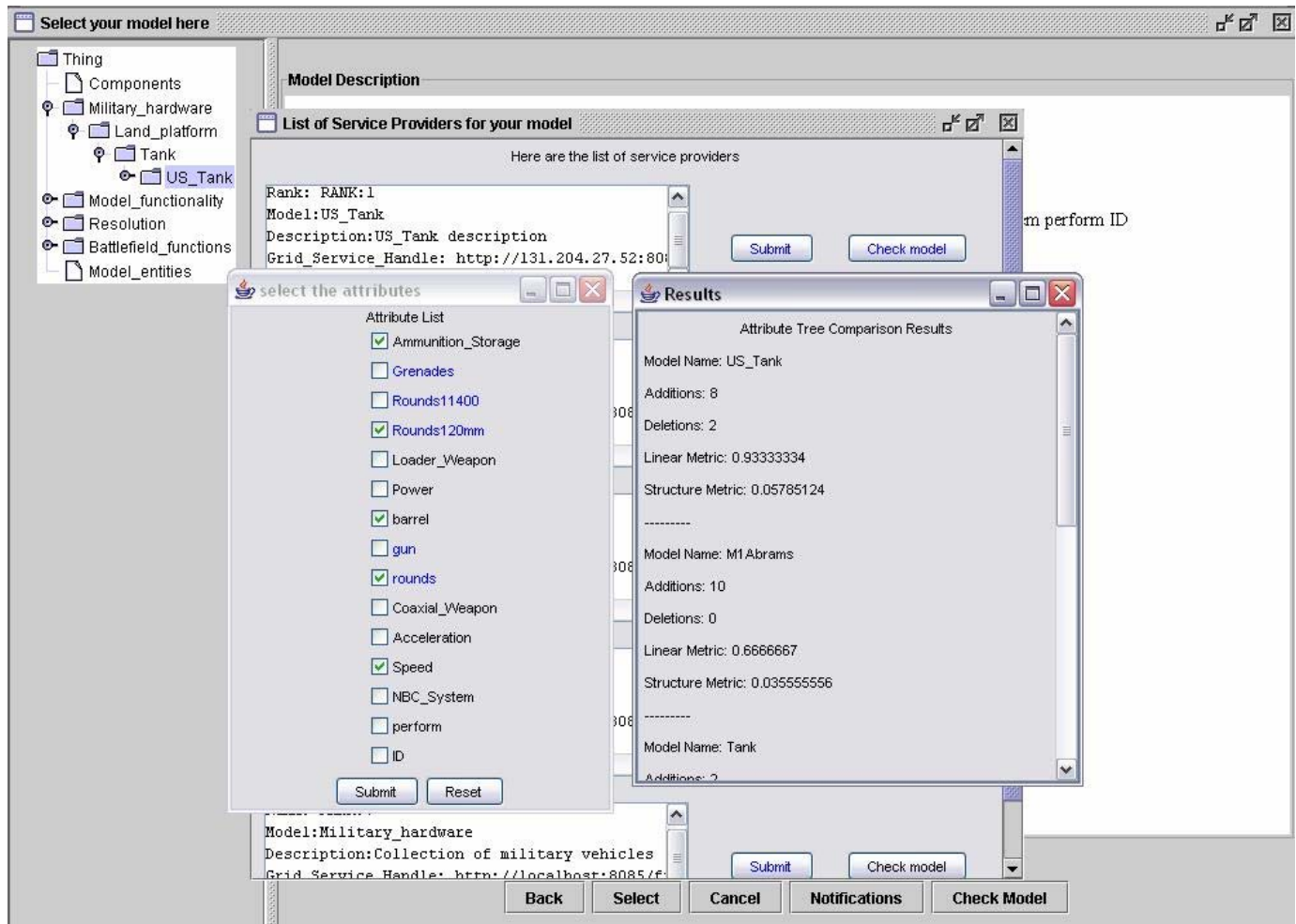


Figure 12. Linear and structural distance (alignment) metrics

5. Conclusions

The separation of interoperation protocols from the simulation infrastructure constitutes the primary contribution of the proposed strategy. The proposed level of indirection via an agent organization aims to decouple the simulation system from the intricate details of instantiation and interoperation of a family of federates to avoid explicit assumptions and to facilitate seamless (run-time) reconfiguration with alternative ensembles. This way, the agent organization abstracts the simulation instantiation and interoperation process. There are two major principles underlying the proposed strategy that makes it useful for the improvement of interoperation. First, the agent organization encapsulates the knowledge about the interoperation administration, alignment, transformation, and management functions discussed in section 2.2. Second, it hides the details about which simulation services are discovered, located, and instantiated as the simulation unfolds. As such, it has the potential to make a federated simulation system independent of how federates are created, composed, and represented. While many of the conflicts that exist between disparate simulations can be resolved via man-in-the-loop simulations, providing such an agent technology can help operators focus on mission-critical activities as opposed to routine interoperability problems.

In this work, while we do not suggest a specific interoperation protocol, the design and implementation aspects for broker, mediator, and matchmaker agents are delineated. The role of facilitator agents in bridging the gap between the meta-level agent framework and the federated simulation infrastructure is also discussed. Further research is needed to explore requirements for an infrastructure that aims to support interoperation among simulations in a federated system. That is, the fundamental requirements for administration, management, alignment, and transformation of content need to be delineated for further progress. For instance, CADM is a Common Architectural Data Model that can be used by the mediator in the agent framework to facilitate mappings during service exchange. Federates do not need to be designed to be compliant with the CADM; that is, the meta-level framework aims to assure that the translation can be performed seamlessly. Concomitantly, to facilitate agent-based mediation, potential types of conflicts between the assumptions and obligations of simulations need to be formalized so that proper procedures can be developed. These assumptions and obligations constitute constraints on the behavioral state space, as well as input and output connector behavior. The realization of a specific

interoperation protocol in terms of these agents will be the immediate extension of the proposed work. In particular, concrete mechanisms that designate 1) how simulations join and leave federations via facilitators; 2) what type of interactions among agents are necessary to orchestrate the brokering, matchmaking, and mediation agents for transparent interoperation; 3) what forms of communication are needed for facilitators to interact with each other; and 4) how such protocols can be specified.

Alignment of the strategy with DoD Architecture Framework (DoDAF) products [25] is another area of significant interest. This extension can facilitate the application of our approach to a broad information technology domain that pertains to architectural frameworks of commands, services, and agencies. The DoDAF entails rules and guidance for developing system architectures at different levels of abstraction and views. Operational view defines the activities and the conceptual view of the application domain. The systems view pertains to the physical interconnections, while the technical view focuses on the standards. The techniques discussed in this work are applicable to the logical data model and physical schema specification components (i.e., OV3 – information exchange models) within the operation and systems views of DoDAF, respectively. Furthermore, intelligent agents that are equipped with computationally decidable formal composability analysis techniques presented by Yilmaz [12] can also be used to resolve interoperation issues that are raised at the level of operational and systems transition specifications. Such specifications pertain to system activity sequencing and timing constraints.

6. References

- [1] Davis, K. P., and R. H. Anderson. *Improving the Composability of Department of Defense Models and Simulations*. RAND Corporation, 2003.
- [2] Page, H. E., R. Briggs, and J. A. Tufarolo. "Toward a Family of Maturity Models for the Simulation Interconnection Problem." Paper 04S-SIW-145 In *Proceedings of the Spring Interoperability Workshop*, 2004.
- [3] Hamilton, A. J., and L. Yilmaz. "Modeling Bilateral Interoperability Through Command and Control Architecture." In *Proceedings of the Military, Government, and Aerospace Simulation (MGA'04) at ASTC'04 Multiconference*, 2004.
- [4] Petty, D. M. "Interoperability and Composability." Modeling & Simulation Curriculum of Old Dominion University, *Short Course Presentation Module #9*, Old Dominion University, 2002.
- [5] Harkrider, M. S., and W. H. Lunceford. "Modeling and Simulation Composability." In *Proceedings of the 1999 Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL, December 1999.
- [6] Petty, D. M., and Eric W. Weisel. "A Composability Lexicon." In *Proceedings of the IEEE Spring Simulation Interoperability Workshop*, Orlando, FL, March 2003.

- [7] Yilmaz, L. "On the Need for Contextualized Introspective Simulation Models to Improve Reuse and Composability of Defense Simulations." *Journal of Defense Modeling and Simulation* 1, no. 3 (August 2004): 135–145.
- [8] Tolk, A. "Composable Mission Spaces and M&S Repositories—Applicability of Open Standards." Paper 04S-SIW-009 In *Proceedings of the Spring Simulation Interoperability Workshop*, 2004.
- [9] Lopes, D., and S. Hammoudi. "Web Services in the Context of MDA." 2003 International Conference on Web Services (ICWS'03), Las Vegas, NV, 2003.
- [10] Tosic, V., B. Pagurek, B. Esfandiari, and K. Patel. "On the Management of Compositions of Web Services." Workshop on Object-Oriented Web Services (OOPSLA 2001), Tampa, USA, October 2001.
- [11] Morris, E., L. Levine, C. Meyers, P. Place, and D. Plakosh. *System of Systems Interoperability (SOSI) Final Report*. 2004. <www.sei.cmu.edu/pub/documents/04.reports/pdf/04tr004.pdf>.
- [12] Yilmaz, L. "Verifying Collaborative Behavior in Component-Based DEVS Models." *Simulation: Transactions of the Society for Modeling and Simulation International*, Special Issue: Component-Based Modeling and Simulation 80, nos. 7–8 (2004): 399–415.
- [13] Foster, I. and K. S. Tuecke. "The Anatomy of the Grid—Enabling Scalable Virtual Organizations." *International J. Supercomputer Applications* 15, no. 3 (2001).
- [14] Foster, I., C. Kesselman, J. M. Nick, and S. Tuecke. "Grid Services for Distributed Systems Integration." *IEEE Computer* 35, no. 6 (2002): 37–46.
- [15] Sycara K., S. Widoff, M. Klusch, and J. Lu. "Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace." *Autonomous Agents and Multi-Agent Systems* 5 (2002): 173–203.
- [16] Genesereth, R. M., and S. P. Ketchpel. "Software Agents." *Communications of the ACM* 37, no. 7 (July 1994): 48–55.
- [17] Hyacinth, S. N. "Software Agents: An Overview." *Knowledge Engineering Review* 11, no. 3 (September 1996): 1–40.
- [18] Tolk, A., and S. Y. Diallo. "Model Based Data Engineering for Web Services." Accepted for publication in *IEEE Internet Computing*, 2005.
- [19] Durfee, H. E., D. L. Kiskis, and W. P. Birmingham. "The Agent Architecture of the University of Michigan Digital Library." *IEEE Proc. on Software Engineering* 144, no. 1 (1997):61–71.
- [20] Kuokka, D., and L. Harada. "Matchmaking for Information Agents." In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 672–678, Montreal, Canada, August 1995.
- [21] Kawamura, M. T., T. R. Payne, and K. Sycara. "Semantic Matching of Web Services Capabilities." 67–72 In *Proceedings of the 1st International Semantic Web Conference (ISWC2002)*, 2002.
- [22] Genesereth, R. M. "An Agent-Based Framework for Software Interoperability." 17–23, In *Proceedings DARPA Software Technology Conference*, 1992.
- [23] Yilmaz, L. "Towards a Metric Suite for Discrete-Event Trace Validity." In *Proceedings of the 2003 Summer Computer Simulation Conference*, The Society for Computer Simulation International, San Diego, CA, 2003.
- [24] Romanowski, C. J., and R. Nagi. "On Comparing Bills of Materials: A Similarity/Distance Measure for Unordered Trees." *IEEE Transactions on Systems, Man, and Cybernetics* 35, no. 2 (Part A, 2005): 249–260.
- [25] DoDAF. *DoD Architecture Framework* <<http://www.software.org/pub/architecture/dodaf.asp#tech>> (last accessed on November 2005).

Author Biographies

Levent Yilmaz is assistant professor of Computer Science and Software Engineering in the College of Engineering at Auburn University. Before joining to faculty in 2003 he was a senior research engineer in the Simulation and Software Division of Trident Systems Incorporated, in Fairfax, Virginia. Dr. Yilmaz earned his Ph.D. and M.S. degrees from Virginia Tech. He worked as a lead project engineer and principle investigator for advanced simulation methodology, model-based verification, and simulation reuse technology development efforts. His research interests are advancing the theory and methodology of simulation modeling, agent-directed simulation to explore and understand dynamics of software processes and project dynamics, and education in simulation modeling. He is a member of ACM, IEEE Computer Society, Society for Computer Simulation International, and Upsilon Pi Epsilon.

Swetha Paspuletti is a graduate student of Computer Science and Software Engineering in College of Engineering at Auburn University. She earned her B.S. degree in Computer Science. Her academic and research interests are software modeling and simulation, software engineering, computer networks, and human-computer interaction. She is a member of ACM and IEEE.