Using ODML to Model and Design Organizations for Multi-Agent Systems [⋆]

Bryan Horling and Victor Lesser

University of Massachusetts
Amherst, MA 01003-9264
{bhorling,lesser}@cs.umass.edu

Abstract. In this paper, we introduce a new, domain-independent organizational design representation able to model and predict the quantitative performance characteristics of agent organizations. This representation, capable of capturing a wide range of multi-agent behaviors, can support the selection of an appropriate design given a particular operational context. We demonstrate the capabilities and efficacy of this language by comparing a range of metrics predicted by an ODML model to previously obtained empirical results from a real-world system. We then outline how such models can serve as the foundation for automated organizational design process by modeling a range of organizational possibilities.

1 Introduction

Any real-world system must be tailored to the environment in which it exists if it is to make effective use of the resources and flexibility available to it. In this paper, we will explore the possibility of such tailoring through the system's *organizational design*. The notion of an organizational design is used in many different fields, and generally refers to how members of a society act and relate with one another. This is true of multi-agent systems, where the organizational design of a system can include a description of what types of agents exist, what roles they take on, and how they act both independently and with one another. This additional structure becomes increasingly important as the system scales in number and scope [1]. Imagine how difficult it would be for a large human organization, such as a corporation or government, to function if individuals lacked job descriptions and long-term peer relationships. Agent systems face similar challenges, and can derive similar benefits from an explicit organizational design.

Consider the problem of designing a solution for a complex, resource-bounded domain, such as a distributed tracking system. The system would consist of an array of sensors deployed to track mobile targets. Assume that each sensor is controlled by an agent. Let us further assume that the sensor nodes must collaborate in some way to be successful, because multiple sensors must illuminate a target to correctly obtain its position. Given these assumptions, a designer must determine a way to structure the

[⋆] This material is based upon work supported in part by the National Science Foundation Engineering Research Centers Program under NSF Award No. EEC-0313747. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

agents' behaviors so that tracking may be accomplished. One strategy would create or delegate a single agent to be the *manager* of the entire sensor network. The manager would decide when, where and how each sensor should take measurements, and then process the resulting data to estimate the targets' positions. This layout of responsibilities constitutes a rudimentary organizational design. It specifies what roles agents take on, who they interact with, and where decision making authority is located.

Under some conditions, this simple solution will perform optimally, because the manager can maintain an omniscient view of the entire network's state and use that view to find the best assignment of sensing tasks. However, under real world conditions, where bandwidth and computational power is limited, communication and data processing takes time, and the number of sensors can be arbitrarily large, the weaknesses of this approach quickly become apparent. A different strategy, in the form of a different organizational design, can compensate for these more challenging conditions. For example, we might distribute the manager role among multiple different agents, to more evenly balance the communication and computational loads. We might also create an information dissemination hierarchy among the agents that summarizes and propagates measurement data to use the available bandwidth more efficiently. However, distributing the role can lead to conflicts and lower utility, because no single agent necessarily has the appropriate local context to make the right decision. Similarly, a hierarchical summarization process might introduce unwanted latency and imprecision.

Implicit in this example is the idea that different organizations will affect performance in different ways, either helping or hindering depending on the situation. Intuitively, changing the manner in which agents interact or the pattern that interactions take on can change behavior at both global and local levels. The objectives of a particular design will depend on the desired solution characteristics, so for different problems one might specify organizations which aim toward scalability, reliability, speed, or efficiency, among other things. Confounding the search for such a design is the fact that many potentially important characteristics can be subtle or have complex interactions.

It is our belief that understanding the fundamental causes of these characteristics and developing accurate models of their effects are both critical to selecting an appropriate design. To enable this functionality, we will present a new, domain-independent language designed to capture organizational information in a single predictive structure. This Organizational Design Modeling Language (ODML) incorporates quantitative information in the form of mathematical expressions that are used to predict the characteristics of an organization. Succinct but detailed representations of the range of organizational possibilities can be defined with ODML, using local knowledge to automatically predict the behavior of the global system. This representation addresses the needs outlined above by creating a computational model that is able to uncover the effects of interactions, and allow subtle effects to become more transparent.

Section 2 will define the ODML language, and demonstrate its ability to model a complex, real-world system. Our ultimate objective is to create technology able to reason about the type of characteristics and tradeoffs mentioned above, automatically selecting the appropriate organizational design for a particular operational context. We will show in Section 3 that by incorporating appropriate search strategies, the detailed information present in ODML can serve as the foundation for such a process.

2 Representing Organizations

The practice of structuring an agent system to best suit a set of perceived strengths and weaknesses is not a new concept. A wealth of research has been performed itemizing algorithms and techniques that accomplish this objective. The solution we present differs in that it suggests a more general way to determine which strategy is most appropriate for a given situation, based on the characteristics endowed on a system by the organization it employs. The foundation of this approach is the representation used to capture those characteristics.

An organizational model, as we envision it, serves in several different capacities. At design time, it should be possible to use the structure to create and evaluate not just a single organizational instance, but an entire family of organizational possibilities. At runtime, it should accurately describe the current organization. In both cases, the model must be sufficiently descriptive and quantitative that one can evaluate the organization's effectiveness, and rank alternatives according to some specified criteria. Below, we enumerate the desired capabilities and characteristics the modeling language should possess to satisfy these requirements:

1. Represent a particular organizational structure. This would include roles, interactions and associations (e.g., coalitions or teams). Different flows in the organization, such as communication and resources, should be representable.
2. Represent the range of organizational possibilities, by identifying general classes of organizations and the parameters which influence their behavior. Different elements should be able to be modeled at different levels of abstraction. Identify which characteristics are under deliberate control, and which are derived from external factors.
3. Enable concrete performance predictions and allow deductive analysis by quantitatively describing the relevant characteristics exhibited by the structure, the manner in which those characteristics interact, and the constraints they are affected by. For example, both communication overhead and the effect that overhead has on work load should be representable.

Many different organizational representation schemes have been developed by researchers [2–10]. Nearly all these representations can satisfy the first two points, but none are able to incorporate quantitative knowledge in such a way that concrete predictions along multiple, interdependent dimensions can be made directly from the model itself. In this section we describe a new formalism called ODML that explores how such information can be modeled and used.

Most existing representations fall into one of two categories: either they represent a wide range of organizational characteristics abstractly, or they can capture a smaller set of characteristics concretely. The former are usually good at representing what entities or relationships exist or could exist, but cannot compare alternatives in a quantitative way. The latter may contain quantitative knowledge, but have difficulty relating that knowledge to specific organizational concepts, either because the quantitative information is only indirectly related to the organizational structure or because a separate simulation is required to elucidate the actual performance. This mitigates their usefulness if one is hoping to understand the effects a particular organizational design will have, particularly in response to the needs of a dynamic environment.

For example, OMNI [6] and $\mathcal{M}$OISE$^+$[9] can each capture a greater variety of organizational concepts than ODML, but do so in a largely qualitative way. Work by Matson and DeLoach [4] does dynamically compute the quantitative utility of an organization, but does so using only a single aggregate statistic. Conversely, both SADDE [5] and MIT's Process Handbook [8] can incorporate arbitrary quantitative information, but neither couples this information with the organizational structure in a way that enables one to deduce how the characteristics of one aspect of the design affect another. The representation created by Sims [10] does incorporate quantitative information into a structured organizational model, but we believe ODML's more flexible design can model more situations at different levels of abstraction. For example, although one can model individual agents and roles in ODML, the representation does not require that such elements exist. By modeling these concepts only abstractly or not at all, one can potentially create models of much larger systems without the associated high combinatorics. At the same time, this flexibility can make the design search itself more difficult. Each representation has its strengths and ODML's goal is not to supplant these works, but to demonstrate another approach that makes different tradeoffs. As shown in the following section, ODML does so by incorporating a concrete but flexible set of primitives that can model a range of organizational constructs along with the quantitative characteristics that differentiate them.

We believe the principal benefit of using a quantitative representation is the ability to make rapid but precise predictions about organizational performance. If one views the process of organizational design as a search through the potentially very large space of possible organizational structures, a critical part of that search is the ability to evaluate alternative designs with respect to an expected operational context. We further believe that the utility of an organizational structure should not be restricted to just a single metric, but can be based on many characteristics that may be tightly coupled with one another and the structure itself. ODML models are able to capture both the space of alternatives and the complex characteristics they exhibit in a representation that can make predictions much faster than is possible through simulation. If used as part of a design process, this advantage can allow the evaluation of a greater number of alternatives, which will increase the chance that the most suitable design will be found.

2.1 ODML

The formal, domain-independent definition of an ODML template specification $\mathcal{O}$ is given below. Section 2.2 will give examples of how these features are used in practice.

$$\begin{aligned} \mathcal{O} &= \{\mathcal{N}, H, C, K, M, V\} \\ \mathcal{N} &= \{N_0, N_1, \ldots, N_n\} \\ N_i &= \{t, \bar{p}, I, H, C, K, M, V\} \end{aligned} \tag{1}$$

The bulk of the ODML template specification is made up of the set $\mathcal{N}$ of *nodes*, each of which corresponds to a particular physical or logical entity that might exist in the organization. For example, in the sensor network scenario there would be nodes corresponding to managers, relationships, agents and the environment, among other things. Each node N_i contains a number of elements, defined below:

t The node's $type$. This label must be unique within the set of template nodes that make up the organization.
$N.t = \langle symbol \rangle$
$\forall N, M \in \mathcal{N},\ N.t = M.t \Leftrightarrow N = M$

$\bar{p}$ An ordered list of $parameters$ that must be passed to the node's template when an instance of the node is created. These are analogous to the parameters one might pass to an object constructor. Each parameter is specified with a type and local name.
$N.\bar{p} = [\langle symbol, type \rangle, \ldots]$

I The set of node types that this node has an *is-a* relation with using conventional object-oriented inheritance semantics. If we assume that a node's $I = \{a, b\}$, an instance of the node will also be an instance of a and b, possessing the characteristics of all three node types. Is-a relationships cannot be cyclic, i.e., N cannot have itself as a decedent.
$N.I = \{\langle type \rangle, \ldots\}$
$\forall i \in N.I, N \neq i \wedge N \notin i.I \wedge ...$

H The set of node types that this node has a *has-a* relation with. If we assume that $H = \{a, b\}$, an instance of the node will possess some number of instances of both a and b. It is through this type of relationship that the primary organizational decomposition is formed. Each has-a has a magnitude that specifies the number of instances connected by the relationship.
$N.H = \{\langle symbol, type, magnitude \rangle, \ldots]$
$magnitude = \langle symbol \rangle$

C A set of $constants$ that represent quantified characteristics associated with the node. Constants may be defined with numeric constants (e.g., 42), or mathematical expressions (e.g., $x + y$).
$N.C = \{\langle symbol, expression \rangle, \ldots\}$

K A set of $constraints$. Also defined with expressions, an organization is considered valid if all of its constraints are satisfied.
$N.K = \{\langle symbol, op, expression \rangle, \ldots\}$
$op \in \{<, >, \leq, \geq, =, \neq\}$

M A set of $modifiers$ that can affect (e.g., mathematically change) a value contained by a node. Multiple modifiers may affect the same value. Modifiers model flows and interactions by allowing the characteristics and decisions made in one node to affect those of another.
$N.M = \{\langle symbol, op, expression \rangle, \ldots\}$
$op \in \{+, -, \times, \div\}$

V A set of $variables$, representing decisions that must be made when the node is instantiated. Each variable is associated with a range of values it can take on. For example, a node might have a variable x that could take any one value in the set $[2.7, y^2, \pi z]$.
$N.V = \{\langle symbol, \{expression, \ldots\} \rangle, \ldots\}$

$symbol$ refers to a user-defined string, similar to a variable name in a conventional programming language. These typically describe or refer to a particular characteristic. $type$ is the type name of some defined node, so $\exists N \in \mathcal{N}$ such that $N.t = type$. $expression$ is an arbitrary algebraic expression, possibly referencing constants, symbols and function calls. ODML supports the use of floating point values, lists of floating point values, and discrete probabilistic distributions in these expressions.

The top-level organization node $\mathcal{O}$ also contains the elements H, C, K, M, V, providing a location to embed additional global information and constraints.

Collectively, we refer to C, K, M, V as a node's *fields*, and the quantitative state of a field as its *value*. For example, a constant field *total_load* might be defined with the expression *total_load = work_load + communication_load* and have a value of 0.9 for a particular agent. Note that the use of the term "constant" may initially be misleading. While the expression defining *total_load* is fixed, the value for *total_load* produced by that expression may change through the application of modifiers, or due to changes in fields or values that the expression is dependent on.

At first glance, the ODML language may appear to be devoid of almost all the organizational concepts that are provided by typical organizational representations. This is partially true, and by design. Instead of directly incorporating the usual high-level organizational components, such as hierarchies, roles, agents, etc., ODML provides a set of relatively low-level primitives by which such structures can be defined. For example, a node with the user-defined type *manager*, having a has-a relationship with another node of type *agent* could embody a role-agent relationship. A sequence of has-a relationships between nodes could indicate a hierarchy. Although the high-level semantics for these nodes may only be implicit, the concrete characteristics and design ramifications are still directly and quantitatively captured by the nodes' fields. We feel that this approach can lead to an increased diversity of representable structures, by avoiding the assumptions and inevitable restrictions that typically accompany high-level structures. Section 3 will demonstrate that it also simplifies the organizational search process, by unifying the various ways that design alternatives can be specified into two well-defined template characteristics (has-a relationships and variables).

ODML instances are quite similar to ODML templates. The difference is that where a template is a description of what *could be*, an instance is a description of what *is*. Where a template might specify that a *manager* role can be assigned to a single *agent* or distributed across multiple *agent* nodes, an instance would indicate that *manager_1* is distributed across *agent_5* and *agent_7*, and so on. Once instantiated, the expressions defined by the fields, the data passed in through parameters, and the interactions caused by relationships can all be used to predict values for an individual node's characteristics.

The formal definition of an instance is nearly identical to that given in Equation 1, so we will not repeat it here. The differences principally relate to the replacement of node types in the template with instances of those nodes in the organizational instance. Thus, the set $\mathcal{N}$ is the set of node instances, whose individual types no longer need be unique. So, where there might be just a single $manager$ type in the template, there can be an arbitrary number of $manager$ instances in the instance. Both is-a ($N.I$) and has-a ($N.H$) relationships no longer reference node types, but particular node instances in $\mathcal{N}$. Finally, the set $\bar{p}$ is filled with appropriate values from each node's parent, and

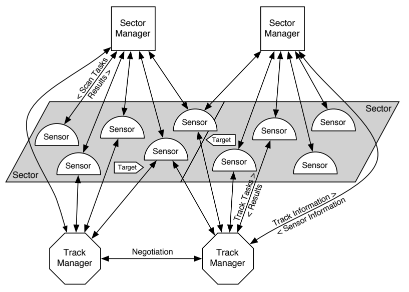
```
**get_value**(*symbol s*)
    $r \leftarrow null$
    **if** ($s$ is of the form $s_1.s_2$)
        $n \leftarrow$ **get_value**($s_1$)
        $r \leftarrow n.$**get_value**($s_2$)
    **else if** ($\exists\, c \in C \mid c.symbol = s$) $r \leftarrow$ **evaluate**($c.expression$)
    **else if** ($\exists\, h \in H \mid h.symbol = s$) $r \leftarrow h$
    **else if** ($\exists\, v \in V \mid v.symbol = s$) $r \leftarrow$ **evaluate**($v.expression$)
    **else if** ($\exists\, p \in \bar{p} \mid p.symbol = s$) $r \leftarrow p$
    **else forall** $i \in I$
        $r \leftarrow i.$**get_value**($s$)
        **if** ($r \neq null$) **break**
    **forall** $m \in M$
        **if** ($m.symbol = s$)
            $r \leftarrow r\ m.op$ **evaluate**($m.expression$)
    **forall** $n \in \mathcal{N}$
        **forall** $m \in n.M$
            **if** ($m.symbol$ is of the form $s_1.s_2$) $\wedge$ ($s_1 = N$) $\wedge$ ($s_2 = s$)
                $r \leftarrow r\ m.op\ n.$**evaluate**($m.expression$)
    **return** $r$

**evaluate**(*expression e*)
    **forall** $s \in \{$ non−function symbols referenced by $e$ $\}$
        $v_s \leftarrow$ **get_value**($s$)
        substitute all occurrences of $s \in e$ with $v_s$
    $r \leftarrow$ mathematical result of $e$
    **return** $r$
```

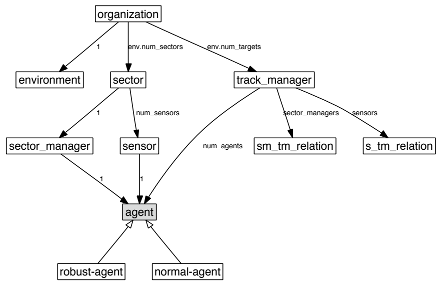
Fig. 1. Pseudocode for the get_value function of a node N. This function is used to quantify the characteristics of instance nodes.

the variable set V for each node is replaced by a single item from that variable's range. Because a common syntax is shared between the two forms, for the remainder of this document I will indicate where necessary which is being considered.

As mentioned above, it is the ability to use an ODML model to deduce quantitative values for specific characteristics that sets it apart from other representations. The manner in which these values are determined for an instance node's characteristics is defined by the pseudocode in Figure 1. This shows how various sources of information, non-local data and node interrelationships all interact to describe the features of a particular node. It is through the execution of this function on a particular symbol that predictions are made of the design's performance. The process shown by the pseudocode formalizes what should be a mathematically intuitive way of determining a symbol's value. For example, if a constant c is defined to be $a.x + y$, then $get\_value(c)$ will proceed by evaluating $a.x + y$, which will recursively call $get\_value()$ on $a.x$ and y. The dot-



(a)



(b)

Fig. 2. The a) conceptual DSN organizational design and b) corresponding ODML template structure.

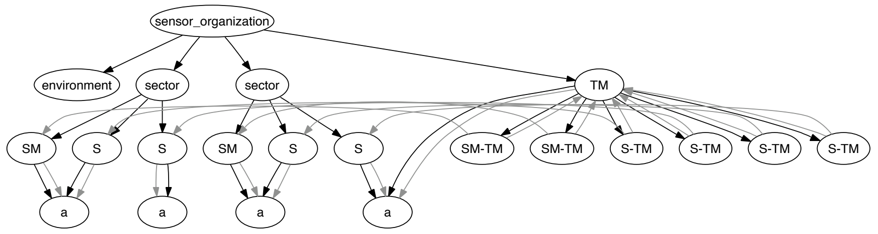


Fig. 3. An instance created from the template in Figure 2b.

notation of $a.x$ indicates that the value of x in node a is being referenced. y refers to the value of y in the local node, which may be another local constant, variable, or parameter, or be inherited from an is-a relationship. The effect of modifiers is somewhat less intuitive, because they may change the value of a symbol without being part of that symbol's definition. For example, there might be a modifier in some other node b that increments the value of c by some amount z. The effect of this operation is as if c were actually defined as $(a.x + y) + b.z$.

Section 3 will show that the process of finding an appropriate organization revolves around first finding the set of valid designs, and selecting from that set the one that is most desirable. The validity of a particular organizational instance $\mathcal{O}$ is defined as:

$$\begin{aligned}&\mathcal{O} \text{ is valid iff } \forall N \in \mathcal{O}.\mathcal{N}, N \text{ is valid} \\ &N \text{ is valid iff } \forall k \in N.K, (N.get\_value(k.symbol)\ k.op\ k.expression) = true\end{aligned} \tag{2}$$

The "desirability" of instance $\mathcal{O}$ can be quantified by defining a $utility$ characteristic in the organization. This can then be computed using the existing machinery by calling $\mathcal{O}.get\_value(utility)$. Once such a value has been computed for all candidate organizational designs, they may be ranked and the best selected. These concepts will be revisited in Section 3.

2.2 Representing Organizational Characteristics

ODML's method of defining organizations is more free-form than most other existing organizational representations. On one hand, this characteristic can offer a great deal of flexibility in how and what organizations are represented. The drawback is its lack of high-level guiding structures, which can make it difficult to initially grasp how ODML models should be created and how the various pieces of a model really interrelate. Because of this, the capabilities of ODML are best explained through an example.

We will use a working system that was developed for a distributed sensor network (DSN) domain to motivate and demonstrate the capabilities of ODML. This framework was designed and developed prior to the existence of ODML, making it an ideal platform to gauge ODML's ability to accurately depict the characteristics of a real-world system. The goal in this domain is to use a collection of sensors to track targets in an area. Each sensor is controlled by an agent that can communicate over a low-bandwidth, wireless network. Individual sensors can return only simple amplitude and frequency values, so a sensor is incapable of determining the absolute position of a target by itself. To track under these conditions, the sensors must be organized and coordinated in a manner that permits their measurements to be used for triangulation.

The system employs an explicit organizational design, as seen in Figure 2a, in an effort to reduce overhead without negatively impacting performance. This begins with the agents dividing the environment into a series of logical *sectors*, each a rectangular portion of the available area. The intent of these divisions is to limit the interactions needed between sensors, to reduce and distribute the overall communication load. There are also three types of responsibilities, or *roles*, that agents may take on: *sector manager*, *track manager* and *sensor*. Each role specifies behaviors, responsibilities and interactions that must be enacted by the agent it is assigned to. Agents can take on multiple roles. The complete architecture, described in detail in [11], has been demonstrated successfully in both simulation and real-world experiments.

We will proceed with an overview of how an ODML model was produced for this system. For clarity, the names of nodes and fields that reside in nodes will be represented in italics. A graphical depiction of some aspects of the ODML template created for the DSN domain can be seen in Figure 2b. Space precludes showing the complete XML-based specification, which is roughly 300 lines long, but a representative portion can be seen in Figure 4. Vertices in the graph, such as *sensor* and *track_manager*, correspond to nodes in the ODML model. Nodes can represent both tangible (e.g. *agent*) and intangible (e.g. *sector*) entities. At the root there exists an *organization* node. The organization node serves as the common root of all other nodes in the structure. This is true even if multiple, otherwise independent structures exist, because they still operate within a common environment (the agent world) that relates them even if no other aspects do. For example, the *sector* nodes are related in a peer-to-peer fashion; no concrete entity exists above to manage or control them (although one could be added to do so). Thus, although the graph-inspired design used by ODML facilitates the modeling of decomposable organizations, it is not limited to depicting this class.

Edges in the graph show relations between nodes. Directed edges with a solid arrow represent has-a relations, and the corresponding label indicates the relation's magnitude. For example, consider the *track_manager* node, which corresponds to the DSN's track

```
1   <node type="track_manager">
2     <param>organization:org,environment:env,[sector]:sectors</param>
3     <is-a>entity</is-a>
4     <has-a name="agent" size="num_agents">agent(env)</has-a>
5     <has-a name="sm_relations">forall(sm, sector_managers):sm_tm_relation(org, this, sm)</has-a>
6     <has-a name="s_relations">forall(s, sensors):s_tm_relation(org, this, s)</has-a>
7
8     <!-- Determine target bounds -->
9     <constant name="uncertainty_radius">5</constant>
10    <constant name="influence_radius">uncertainty_radius + 10</constant>
11    <constant name="target_area">3.14 * influence_radiusˆ2</constant>
12
13    <!-- Calculate requested measurement rate -->
14    <constant name="desired_sensors">3</constant>
15    <constant name="sensor_density">forallavg(sectors.sensor_density)</constant>
16    <constant name="actual_sensors_available">target_area * sensor_density</constant>
17    <constant name="requested_sensors">min(desired_sensors, actual_sensors_available)</constant>
18    ...
19  </node>
20
21  <node type="s_tm_relation">
22    <param>organization:org,track_manager:tm,sensor:s</param>
23
24    <!-- Calculate actual measurement rate -->
25    <constant name="requested_sensor_rate">tm.requested_sensors / org.total_sensors</constant>
26    <constant name="requested_measurement_rate">tm.requested_measurement_rate * requested_sensor_rate</constant>
27    <modifier name="s.requested_measurement_rate" op="+">requested_measurement_rate</modifier>
28
29    <!-- Assign measurement communication load -->
30    <constant name="actual_measurement_rate">requested_measurement_rate * s.actual_measurement_ratio</constant>
31    <modifier name="tm.actual_measurement_rate" op="+">actual_measurement_rate</modifier>
32    <modifier name="s.message_rr" op="+">actual_measurement_rate</modifier>
33    ...
34  </node>
```

Fig. 4. A portion of the ODML specification for the *track_manager* and *s_tm_relation* nodes.

manager role. It has a number of *agents* defined by *num_agents*, shown in line 4 of Figure 4, that represent the agents the role is bound to. In the DSN system, a track manager role may migrate among many different agents over time. The magnitude *num_agents* is used here to represent this behavior, and modifiers distribute the characteristics accordingly. Has-a relationships in the template may also be recursive or self-referential. This facilitates the modeling of self-similar organizations, such as hierarchies, by making it possible to represent organizations with varying numbers of levels.

A hollow-arrow edge represents an is-a relation, so *normal_agent* is an instance of *agent*. Shaded nodes, such as *agent*, are abstract and cannot be directly instantiated. Thus, any node with a has-a relation with *agent* can instead substitute *normal_agent*. This indirection allows the model to represent alternative nodes with different capabilities. For example, suppose there were two types of agents available: a normal agent, and a “robust” agent that had better processing and computational powers but a higher cost. To model this, a *robust_agent* node is depicted that also has an is-a relation with *agent*, and can be substituted for *agent* in the same way. A similar arrangement could model a range of alternative roles that had different characteristics but could serve overlapping purposes. For example, if different sensors types existed with different characteristics, they would all have an is-a relationship with the common node *sensor*.

Figure 3 shows one particular instance created from the template in Figure 2b. Vertices in the instance graph represent nodes, and a gray directed edge indicates the existence of a non-local modifier from the source node to a field in the target node. Black directed edges represent has-a relationships, but unlike the template they have no magnitude. Because this is a particular instance of the sensor network organization, the decision points present in the template have all been decided. Therefore, where *sector* might have the *num_sensors* magnitude on its sensor relationship in the template, a discrete value of two has been chosen for that field in this particular instance. Because of this, each sector in the instance has a has-a relationship with two distinct sensors (S). Normal agents (a), sector managers (SM), track managers (TM), and two kinds of track manager relations (SM-TM and S-TM), are also present.

The heart of any ODML model exists in the expressions encoded within nodes' fields. Each expression consists of a sequence of standard mathematical operations (e.g., $+$, $\div$, x^y, etc.) and a limited number of predefined functions (e.g., *min, max, sqrt, round, forallavg*, etc.). A selection of these fields, contained by the *track_manager* and *s_tm_relation* nodes, are shown in Figure 4. The former defines the track manager role, while the latter represents the relationship that role has with sensors in the environment. Each node's field contains a mathematical equation, combining local and nonlocal information to calculate new local values. These expressions allow one to represent how different characteristics of the node may be computed. For example, suppose we wish to define how to calculate the bounds of a target that moves through the environment. This will depend on the uncertainty the manager has in the target's location, along with a factor modeling the range of the target's influence. In our model, this area will be approximated as a circle; line 11 shows how the track manager's *target_area* constant is derived from the target's *influence_radius*. The number of sensors presumed capable of sensing the target is the average number that lie within this area. Therefore, although the number of *desired_sensors* is independent of the environment, the *actual_sensors_available* to the manager depends indirectly on the *target_area* and *sensor_density*, as shown in line 16. The constant *requested_sensors* is the minimum of the desired and available.

We may model the number of measurements provided to the track manager in a similar way. The *actual_measurement_rate* constant from line 30 of the sensor-track manager relationship is derived from the locally calculated *requested_measurement_rate* and *actual_measurement_ratio* computed by the sensor node. This value is then used in a pair of modifiers defined in lines 31 and 32 that specify for the track manager and sensor the actual number of measurements that will be taken. This value will eventually be used along with other characteristics to estimate the *rms_error* the track manager will exhibit. Variables and constraints present elsewhere in the model are similarly defined. These descriptions give only the spirit of the calculation process, refer back to Figure 1 for a more rigorous definition of how these values are assimilated in practice.

Through modifiers or the assimilation of nonlocal values, the characteristics of one node may affect or be affected by those of another. ODML models are generally constructed by designing individual nodes, and linking them through nonlocal dependencies or modifiers. The resulting web of equations allows one to model important concepts such as information flow, control flow, and the effects of interactions. By propagating data through these expressions, the model can correctly predict the characteristics of both individual nodes and the organization as a whole. Perhaps more importantly, it also allows the model to predict characteristics not envisioned by the designer, as results can flow through the graph in unanticipated ways.

Although space precludes providing a complete description, the model we have created also captures a range of other characteristics of the DSN system, including the physical and task environment, agent interactions, single and multiple role assignments,

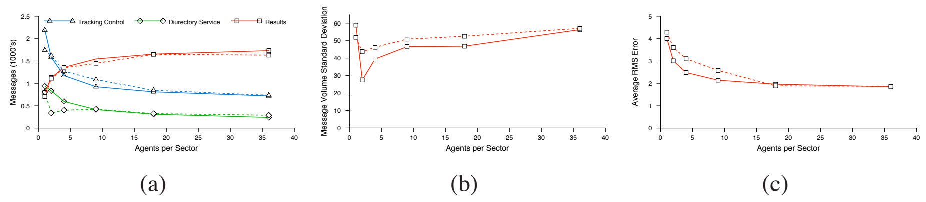


Fig. 5. ODML DSN model predictions versus empirical observations for a) Message totals by type, b) Messaging disparity and c) RMS error. Predicted lines are solid, empirical are dashed.

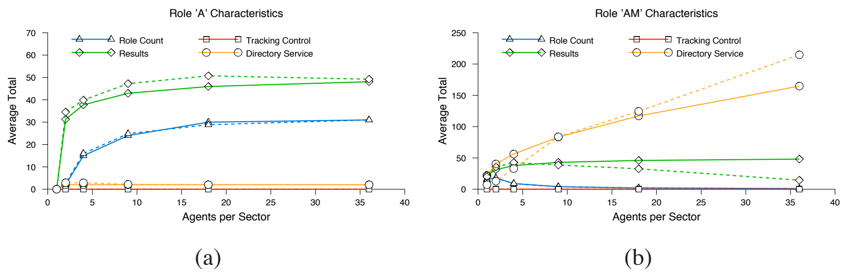


Fig. 6. Comparison of ODML DSN model's role-specific predictions.

dynamic role assignment, heterogeneity, geographic coalitions, potential conflicts, and both hard and soft constraints. Each was successfully modeled, suggesting that the relatively modest set of primitives offered by ODML is capable of representing a wide range of complex and relevant organizational factors.

One feature absent in the ODML language is a well-defined notion of time. Although the DSN model captures rates of change and expected value, there is no explicit representation of a varying timeline or change points present in ODML. This makes ODML more tractable to work with by reducing the search space of organizations, but can also make representing some characteristics more difficult. The absence of time means that ODML instances generally represent a snapshot of a running system, or an averaging or probabilistic distribution of effects as they would occur over some span of time. If dynamic elements exist, they may be represented in that way, and can still be used to guide organizational decisions. For example, one might identify that sensor resources must be available to satisfy the needs of dynamically discovered targets. By using an expected or worst-case value to quantify this need, or creating a distribution that captures the range of possible targets, the organization can be designed appropriately even though the specific context is uncertain at design time.

2.3 Evaluating the Representation

Previous work analyzed the effects that the DSN organization had on performance across a range of metrics [12]. In those tests, the number of agents in each sector was varied to demonstrate how changing the organization can have far-reaching consequences. To gauge the representational efficacy of ODML, we have used the model described in the previous section to create organizational instances that match those prior test runs. Characteristics defined in the ODML model measure the different metrics that were originally tested, allowing us to calculate values that can be compared against the empirical results. Values for these characteristics were obtained using the get_value function described in Figure 1. This both demonstrates how ODML can be used as a predictive tool for different operating contexts, and evaluates how well a specific model was able to capture real-world behaviors. Ideally, the empirical and modeled results will match, demonstrating that the model captures the complexity present in the system and that predictions derived from the model are accurate.

The comparative results are shown in Figure 5. Note that the behavioral details behind these results, which are presented in [12], are beyond the scope of this document. In this context, we are exploring only the accuracy of the ODML model's predictions. Solid lines in the graphs represent the values predicted by the ODML model, while dashed are those obtained through the previous empirical testing. Figure 5a shows communication totals by type. Figure 5b shows the communication disparity, which measures how well or poorly the communication load is distributed in the population. Figure 5c shows the average RMS error of the tracking tasks. Although there are some points of difference, in most cases the model does a good job predicting performance. One difference can be seen in Figure 5b, where the predicted standard deviation underestimates the actual performance in most cases. This is a byproduct of our assumption that all sensors were equally used. In the running system, sensors in the center of the environment are used more than those at the edges, and will have different communication profiles because of it. Our model does not capture these geographic differences, and will therefore generally have a lower estimated deviation.

To evaluate how our model captures finer-grained details, we analyzed the predicted communication profiles of individual roles. A comparison of these role-specific behaviors can be seen in Figure 6. In addition to communication totals, these graphs also include role counts, indicating how many agents take on the specified role. 'A' represents the *sensor* role, 'M' is the *sector_manager*, while 'T' is the *track_manager*. 'AM' describes agents acting as both sensors and sector managers. Predictions at this more detailed level are also accurate. Many of the differences that do exist can be attributed to geographic variances in a small sample size. For example, the 36- and 18-size scenarios had only one or two sector managers. Their individual geographic locations would likely affect performance, and these variations are not reflected in the predicted values.

3 Using ODML to Design Organizations

We have thus far argued that organizations can have a tangible effect on performance, and it is therefore useful to be able to understand those effects when designing an agent system. The previous section demonstrated ODML's ability to correctly model and predict the global and local characteristics of particular organizational designs used in a previously constructed, real-world application. Our ultimate goal, however, is to use this capability to support an automated organizational design process. It is this objective that motivates the need for the detailed information that makes up an ODML model. In this section we will outline how this has been accomplished, by using ODML as the foundation for a new design methodology. Space permits only a superficial description of this work, a more detailed description will be given in a later publication.

Recall that ODML representations are divided into two distinct classes: *templates* that encompasses the range of all possible organizations, and *instances* that are each particular organizations derived from a template. The range of possible instances in a template, defined by how its variables and has-a relations can be decided, constitutes an organizational space that can be searched.

For example, one of the *environment* node's variables expresses the range of possible *sensors_per_sector*, which controls the shape of part of the organization. Other uses of variables might be to decide the relative priority of an agent's tasks, or the amount of time it is willing to wait for a response. Decisions made for the *agent* has-a relationships in the three roles will determine the specific role-agent bindings that will be used. Sequences of similar decisions could also decide if the manager role will be distributed, or how tall a data processing hierarchy should be. A range of options are possible, and different choices will result in different organizations. Constraints defined within nodes have the opposite effect, by limiting the set of valid organizations to some subset of those that are possible.

As with other characteristics, an ODML model can be used to capture the expected *utility* of an organization by relating organizational characteristics with mathematical expressions. For example, in the DSN model, *utility* is based on *average_rms*, which is derived from the *rms_error* values of the *tracker* roles, which are themselves dependent on many other factors. By defining a field with the semantics of "utility", one can quantitatively evaluate and rank candidate organizations by comparing these fields' values. This provides a clear way to discriminate and decide among the alternatives that exist in the template's organizational space.

An automated organizational design process can be built upon these two concepts, by leveraging the predicted characteristics to navigate the organizational space. Each ODML model includes a description of the expected operating context, available resources, etc., in addition to the possible organizational structures. Given such a model, the most appropriate design (i.e. the one with maximal *utility*) can be found with an appropriate search of the organizational space defined by the model.

The number of possible organizations grows exponentially with the number of decisions that must be made in its construction. When organizations are being designed for a large number of agents, or when the template is particularly flexible, the space of possibilities can easily become intractable. Because of this, it is important to develop techniques able to cope with such situations.

One technique that we have implemented bounds the search by exploiting hard constraints that exist in the system. The expression-based knowledge in the ODML model makes this possible. Recall that all hard constraints must be satisfied in an organization for it to be considered valid (see Equation 2). If a constraint has become unsatisfied during the course of an organizational search, it may be reasonable to halt the search before the organization is fully formed, and backtrack from that point. Two issues complicate this process. The first is that constraints may be initially unsatisfied in a partially instantiated structure, and only become satisfied later in the decision making process. Additionally, because values may change nonmonotonically, a constraint can change its state repeatedly during the instantiation process. Both cases may cause a backtrack decision to be incorrect by missing valid organizations.

By analyzing the relationships that exist between fields, it is sometimes possible to determine when choices may be safely ignored. For example, in the DSN domain a single agent cannot control more than one sensor. Each agent has a *sensors_controlled* value, that is initially zero and later incremented using a modifier when it is assigned to a sensor. This restriction is modeled by the hard constraint *sensors_controlled* ≤ 1. For a particular organization with n sensors and a available agents, there are a^n possible assignments of agents to sensors. However, only $\binom{a}{n}$ of them are valid according to the *sensors_controlled* constraint. If it were possible to detect when an invalid assignment had been made before all organizational decisions have been completed, one could bound the search at that point and backtrack to where the constraint was satisfied.

Because role assignments are permanent, there is no decision that could be made to reduce the number of sensors controlled by an agent. Therefore, it is reasonable and correct to bound the search and backtrack if an agent is ever found to control more than one sensor, because that constraint will never be satisfied. While a human expert might intuitively know to avoid such impossible configurations, a generic search process is too myopic to perceive this fact. However, the relationship that this knowledge is based upon is represented in the organizational template, in that *sensors_controlled* is affected by only that one type of modifier from the *sensor* node, that increments the value by a positive constant. It is possible to use this information to deduce the monotonically increasing trend of *sensors_controlled*, and backtrack when appropriate.

Our technique uses partial derivatives on a field's dependent variables as a general way of determining the trend of that field's value. The trends for a constrained value and the expression it is constrained by are determined by first recursively finding the trends of all the fields they depend on. By taking the partial derivative with respect to each field referenced by an expression, along with the trend of each field, the trend of the original expression may be determined. With this information one can automatically detect when constraints have become unsatisfiable, and correctly bound the search.

4 Conclusions

The ODML-based approach that we have presented in this paper is both general and flexible enough to model a range of common organizational characteristics. It is particularly useful when describing features that have a quantitative character. To support this claim, we have created and tested a complete model of a previously existing sensor network framework. Other work has also used ODML to model a hierarchical information retrieval service [13], as well as more abstract, theoretical problems such as SUBSET-SUM and TILINGS. These models demonstrate ODML's ability to accurately predict the small- and large-grained behaviors of an organizationally-driven agent system.

We believe the detailed, quantitative knowledge embedded in ODML models is relevant because it provides the foundation for prescriptive technologies, such as the automated design service outlined above. We have described one strategy currently used to cope with the potentially large organizational search space by inferring value trends to find unsatisfiable constraints. Additional techniques are currently being developed. Armed with such techniques, the full potential of ODML can be realized by using its flexible but detailed representation of organizational possibilities to effectively design agent organizations.

References

1. Corkill, D.D., Lander, S.E.: Diversity in Agent Organizations. Object Magazine **8** (1998) 41–47
2. Tambe, M., Adibi, J., Alonaizon, Y., Erdem, A., Kaminka, G.A., Marsella, S., Muslea, I.: Building agent teams using an explicit teamwork model and learning. Artificial Intelligence **110** (1999) 215–239
3. Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., NagendraPrasad, M., Raja, A., Vincent, R., Xuan, P., Zhang, X.: Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. Autonomous Agents and Multi-Agent Systems **9** (2004) 87–143
4. Matson, E., DeLoach, S.A.: Autonomous organization-based adaptive information systems. In: Proceedings of the IEEE International Conference on Knowledge Intensive Multiagent Systems (KIMAS ’05). (2005)
5. Sierra, C., Sabater, J., Augusti, J., Garcia, P.: SADDE: Social agents design driven by equations. In Bergenti, F., Gleizes, M., Zambonelli, F., eds.: Methodologies and software engineering for agent systems. Kluwer Academic Publishers (2004)
6. Dignum, V., Vazquez-Salceda, J., Dignum, F.: Omni: Introducing social structure, norms and ontologies into agent organizations. In: Second International Workshop on Programming Multi-Agent Systems at the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, New York, NY (2004) 91–102
7. Fox, M., Barbuceanu, M., Gruninger, M., Lin, J.: An Organizational Ontology for Enterprise Modeling. In Prietula, M.J., Carley, K.M., Gasser, L., eds.: Simulating Organizations: Computational Models of Institutions and Groups. AAAI Press / MIT Press (1998) 131–152
8. Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C.S., Bernstein, A., Herman, G., Klein, M., O’Donnell, E.: Tools for inventing organizations: Toward a handbook of organizational processes. Management Science **45** (1999) 425–443
9. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Proceedings of the Brazilian Symposium on Artificial Intelligence (SBIA’02). (2002) 118–128
10. Sims, M., Corkill, D., Lesser, V.: Separating Domain and Coordination in Multi-Agent Organizational Design and Instantiation. In: Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004), Beijing, China (2004)
11. Lesser, V., Ortiz, C., Tambe, M., eds.: Distributed Sensor Networks: A Multiagent Perspective (Edited book). Volume 9. Kluwer Academic Publishers (2003)
12. Horling, B., Mailler, R., Lesser, V.: A Case Study of Organizational Effects in a Distributed Sensor Network. In: Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004), Beijing, China (2004)
13. Horling, B., Lesser, V.: Quantitative Organizational Models for Large-Scale Agent Systems. In: Proceedings of the International Workshop on Massively Multi-Agent Systems, Kyoto, Japan (2004) 297–312