Referent Tracking for Command and Control Messaging Systems

Shahid Manzoor, Werner Ceusters and Barry Smith
Center of Excellence in Bioinformatics & Life Sciences, University at Buffalo
701 Ellicott Street, Buffalo NY, 14214
smanzoor@buffalo.edu, ceusters@buffalo.edu, phismith@buffalo.edu.

***Abstract* – The Joint Battle Management Language (JBML) is an XML-based language designed to allow Command and Control (C2) systems to interface easily with Modeling and Simulation (M&S) systems. While some of the XML-tags defined in this language correspond to types of entities that exist in reality, others are mere syntactic artifacts used to structure the messages themselves. Because these two kinds of tags are not formally distinguishable, JBML messages in effect confuse data with what the data represent. In this paper we show how a realism-based ontology combined with a rule language can be used to make these distinctions explicit. The approach allows storage of the contents of JBML messages in a Referent Tracking System in a format that mimics the structure of reality thereby providing an aid to message validation.**

***Index Terms* - Realism-Based Ontology, Command and Control, Referent Tracking, JBML**

I. Introduction

The Joint Battle Management Language (JBML) was designed in response to the growing need to interface Modeling and Simulation (M&S) systems with Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) systems in order to provide the user with capabilities that support operational functions including Course of Action (COA) Development and Analysis, mission rehearsal, and execution monitoring [1]. Its main benefit is that it overcomes the interoperability problem caused by the use of unstructured "free text" within the operational C2 messages that are passed within C4ISR systems, specifically for the most critical C2 information: the commander's intent, orders and directives. While suitable for interpersonal communication, free text is inadequate for use by simulations and robotic or software-based components.

JBML is highly structured and the majority of its XML-tags (or '*elements'*) are given very intuitive labels. Yet, the language generally suffers from the problem that while some of the tags denote generic entities in reality that are instantiated by the referents denoted by the elements' contents in a specific message, other tags provide information about the message structure itself, rather than what the message is about. Because these two kinds of tags are not formally distinguishable, JBML messages confuse data with what these data represent. This does not need to be a problem if the data are always an accurate representation of reality. But in an intelligence context, where large amounts of data are uncertain, conflicting and erroneous, either because of misinterpretations committed by friendly agents or falsifications introduced through counter-intelligence activities, this distinction is crucial.

In this paper we show how a realism-based ontology, combined with a rule language, can be used to make these distinctions explicit and to store the contents of JBML messages in a Referent Tracking System (RTS) in a format that mimics the structure of reality.

In [2] we demonstrated how an RTS, i.e. a system that forces assertions to be expressed by means of statements that use globally unique and singular identifiers for each referent mentioned [3] and that are structured according to the principles outlined in Basic Formal Ontology [4], may help the Intelligence Community to share information unambiguously. Here, we report on further enhancements of our prior work to integrate XML-based information systems with RTS, using JBML as a specific example. We describe a prototype application that uses a realism-based ontology to parse JBML messages and that provides reasoning facilities for the validation of the data. The application works by (1) generating identifiers for all the real-world entities and their relationships that are mentioned in given messages, and (2) storing these identifiers in an RTS.

II. Ontology-based processing of military messages

An RTS is an ontology-based database which uses globally unique identifiers (called IUIs) to represent: (1) instances (spatiotemporal particulars, including people, commands, places, events), (2) the relations that hold between such instances, (3) the (potentially multiple) names that are assigned to such instances, and (4) the universals or classes that such instances instantiate [2, 3, 4]. The RTS thereby provides a framework for the logically coherent formulation of assertions made about any of the given entities, including those types of assertions contained in C2 messaging.

An example of a military message in JBML is shown in Listing 1. The message represents a mission that is assigned to an army unit with identifier '2TF A TEAM'. The army unit has to move between points whose coordinates are given under the WhereValue element in the XML. In the prototype implementation here described, assertions about entities are inserted in the RTS as triples, for example:

IUI-1004	instanceof	MilitaryMission
IUI-1005	instanceof	MilitaryUnit
IUI-1004	missionAssignedTo	IUI-1005
IUI-1005	currentPostionAt	IUI-1014

The first triple represents a statement to the effect that the entity IUI-1004 instantiates the universal MilitaryMission; the second triple a statement to the effect that the entity IUI-1005 instantiates the universal MilitaryUnit; the third triple represents a statement to the effect that the mission with identifier IUI-1004 is assigned to the military unit with identifier IUI-1005. The fourth triple represents a statement to the effect that the current position of IUI-1005 is location IUI-1014.

We have developed a prototype ontology for command and control systems called 'C2-Test-Ontology', which uses a subset of UCore SL, an ontology designed to support the Universal Core data schema [5], to link the JBML tags from the military messages to UCore SL. The ontology tells us what data elements in a military message refer to what kinds of real world entities (e.g. ArmyUnit, MilitaryMission) and what relationships obtain between them.

We have also developed a Middleware program which parses military messages, and – drawing on knowledge contained in the C2-Test-Ontology – communicates with the RTS and a rule-based reasoner to assign IUIs to the entities mentioned in the messages. The architecture of our system is shown in Fig. 1. When the middleware runs for the first time, and thus no messages have as yet been processed, the RTS database contains no information. As more and more messages are processed, the RTS becomes populated with successively more IUIs and with corresponding statements about the relationships between the referents of these IUIs. Knowledge obtained through parsing earlier messages is used to improve parsing of new messages, to check the consistency of the information provided in new messages, and so forth.

```
<OrderPush>
   <Task> <GroundTask>
      <TaskeeWho>
         <UnitID>2TF A TEAM</UnitID>
       </TaskeeWho>
      <What><WhatCode>MOVE</WhatCode></What>
      <Where>
      <AtWhere>
         <JBMLAtWhere>
            <WhereValue>
               <WhereLocation Sequence="0">
                     <GDC>
                           <Longitude>48.9100583</Longitude>
                           <Latitude>39.9570562</Latitude>
                           <ElevationAGL>0.0</ElevationAGL>
                      </GDC>
                  </WhereLocation>
                  <WhereLocation Sequence="1">
                     <GDC>
                        <Longitude>48.9056792</Longitude>
                        <Latitude>39.9891024</Latitude>
                        <ElevationAGL>0.0</ElevationAGL>
                     </GDC>
                  </WhereLocation>
               </WhereValue>
            </JBMLAtWhere>
         </AtWhere>
      </Where>
   </GroundTask></Task>
</OrderPush>
```

Listing 1: An Example of a JBML Message

A. Step 1: Structural Analysis of XML

In the first step, a military message is parsed with the goal of detecting entities referred to and the relations between them, as shown in Fig. 2. The nodes of the graph correspond to putative entities within the domain of the message, while the edges correspond to relations. Nodes *without* a prefix are entities named explicitly in the message. Nodes with the "rts:" prefix are generated on the basis of the XML-structure of the message. Some of them will correspond to real entities, but others are inserted as a side-effect of the oddities of the XML syntax. It is one of the tasks of our application to turn the resultant distorted view of reality into a realistic picture which conforms to good ontological principles [6].

In the first step, the middleware does not use any knowledge from the UCore SL or C2-Test ontologies. Rather it iterates over the XML structure of the message using a depth first iteration strategy, in which each XML element is viewed as a relation between possible entities. The iteration leads to the creation of a new graph in which the software creates nodes automatically without contacting the RTS. As an example, the XML element '<Task>' is not explicitly labeled in the message above. It is immediately followed by <GroundTask>, which is also not explicitly labeled. What is the relation between these two elements? The message provides no answer to this question. Our application in this first step assigns IUI identifiers to Task and GroundTask. By using ontologies in later steps, it will become possible to arrive at more determinate representations.

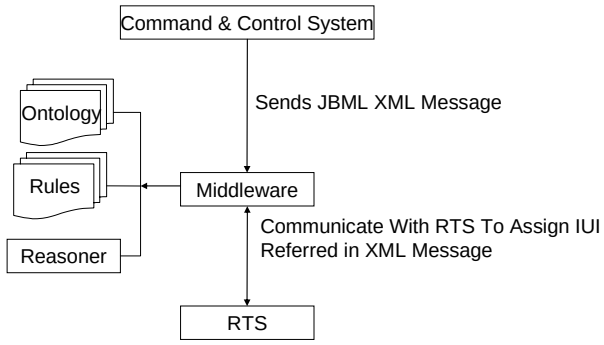

Fig. 1: Command & Control Application Integration with RTS

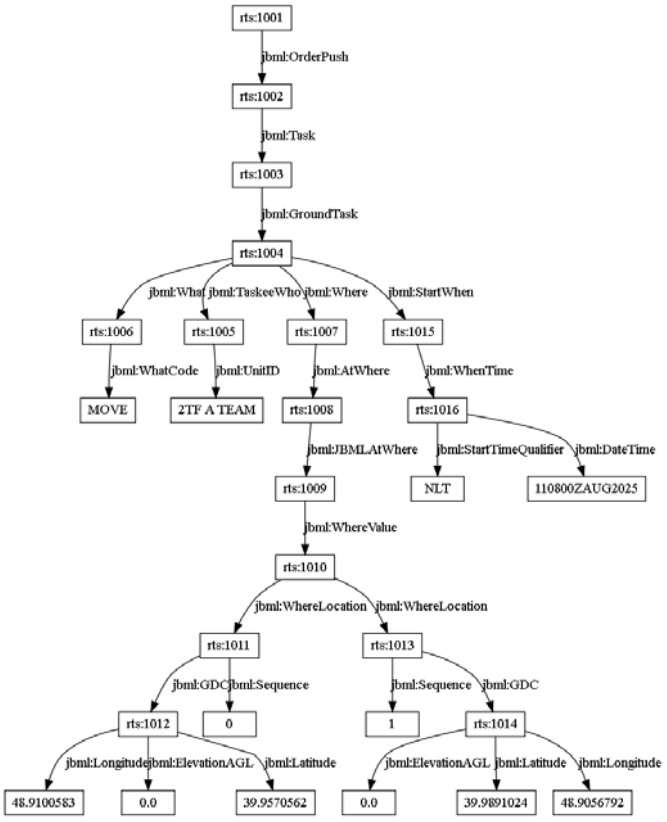

Fig. 2: Graph generated in the first step of the middleware

The entities and their relations shown in the graph in Fig. 2 are represented in the middleware by means of triples. The first triple in the graph is:

rts:1001 jbml:OrderPush rts:1002

The first and last terms of this triple denote entities, while the middle term denotes the relationship between these entities.

As a further example, the triples representation for the node rts:1004 and its relations with rts:1015 and its descendents are shown in Listing 2.

This graph does not yet provide a representation which is faithful to reality in terms of the logical and ontological principles on which UCore SL and our C2-Test-Ontology are based. Clearly, some nodes in the figure denote genuine entities, including:

rts:1004, a particular military ground mission
rts:1005, a particular military unit,
rts:1015, a particular time
rts:1012, a particular location.

Moreover some nodes are such that they are *unique* instance identifiers, as is required by Referent Tracking principles. However, the graph also contains many violations of these principles. Some nodes denote in a non-unique way; for

rts:1004	jbml:StartWhen	rts:1015
rts:1015	jbml:WhenTime	rts:1016
rts:1016	jbml:StartTimeQualifier	“NLT”
rts:1016	jbml:DateTime	“110800ZAUG2025”

Listing 2: Triples representation generated while analyzing the message in Listing 1

example ‘rts:1015’ and ‘rts:1016’ denote the same time, and ‘rts:1011’ and ‘rts:1012’ denote the same particular location. In line with the Referent Tracking principles, such duplicate identifiers must be removed. Inspection reveals that some triples, such as

rts:1001	jbml:OrderPush	rts:1002
rts:1002	Task	rts:1003
rts:1003	jbml:GroundTask	rts:1004

are defective from an ontological point of view and these too must be removed. A task, for example, is clearly not a relationship. This restructuring cannot however be achieved computationally in this step, because the XML framework does not provide explicit information sufficient to ensure the ontological soundness of the representation. We have therefore built another service, executed in step 2 below, which translates information implicit in the XML structure of a message into explicit assertions.

B. Step 2: Shallow translation of XML

To achieve ontologically correct information of this sort we use a variant of the Jena rule-based reasoning language [7] to formulate rules stating the conditions for adding or removing assertions in the graph.

We employ for this purpose forward-chaining reasoning rules, each consisting of (1) a body and (2) a head. The former specifies the set of patterns which triples in the message graph must match if they are to be capable of being transformed into triples that are ontologically well-formed. The latter consists of a set of ontologically well-formed triples into which the triples in the set specified in (1) to be translated.

We distinguish four different types of rules:

- If a rule name is prefixed with ‘ded_’ (for *deduction*), then it is a deduction rule as described above.
- If a rule name is prefixed with ‘red_’ (for *reduction*), then the rule removes the triples matched against the pattern in part (1) of the rule and adds the triple set in part (2).
- If a rule name is prefixed with ‘map_’ (for *mapping*) then it assigns appropriate IUIs to entities that are mentioned repeatedly in the message.

- If a rule name is prefixed with ‘val_’ (for *validation*), then its goal is to check for inconsistencies in the data and to provide corresponding notification to the receiver of the message.

Our middleware provides distinct services for each of the rule types described above.

An example of a deduction rule is:

```
[ded_1 (?x jbml:Task ?y) (?y jbml:GroundTask ?z)
    -> (?z ro:instanceof c2:MilitaryMission)]
```

which asserts that if (i) for two putative entities the ‘Task’ relationship holds and if (ii) the target-entity of this relation stands in a ‘GroundTask’ relation with a third entity, then (iii) the first two entities are erroneous artifacts generated in step 1, while the third entity is a MilitaryMission in the terms of our C2-Test-Ontology. Note that the issue here is not whether this specific rule is correct. What we are offering here are examples. Crafting the needed set of rules for JBML and other XML-based languages is a task which needs to be performed when once the general strategy has been formulated and tested.

Here ‘ded_1’ is the name of the rule. Variables are prefixed by ‘?’ and serve as placeholders for the actual terms in the matched triples in their respective places. The following are triples that match the conditions of the rule above:

(rts:1002 jbml:Task rts:1003) (rts:2003 jbml:GroundTask rts:1004),

where variables in the rule bind terms in the matched triples as follows: ?x = rts:1002, ?y = rts:1003, ?z = rts:1004.

Under the *head* part of the rule, the software inserts the triple ‘rts:1004 ro:instanceof c2:MilitaryMission’, where the term ‘ro:instanceof’ denotes the instanceof relation from the Relation Ontology (‘ro’) [8], and the term ‘c2:MilitaryMission’ denotes the MilitaryMission universal taken from the C2-Test-Ontology (‘c2’).

An example of a reduction rule is:

```
[red_7: (?x jbml:StartWhen ?y)
  (?y jbml:WhenTime ?z)
  (?z jbml:StartTimeQualifier "NLT")
  (?z jbml:DateTime ?l)
  ->    (?x c2:StartWhen ?y)
     (?x ro:instanceof c2:TimeEvent)
     (?x c2:NoLaterThan ?l)]
```

The body of this rule is matched by the following set of triples, generated from the message shown in Listing 2 .

```
rts:1004  jbml:StartWhen          rts:1015
rts:1015  jbml:WhenTime           rts:1016
rts:1016  jbml:StartTimeQualifier “NLT”
rts:1016  jbml:DateTime           “110800ZAUG2025”
```

Since the rule is of type ‘reduction’, the four triples are removed from the military message triple set, and the following three triples are inserted instead:

```
rts:1004 c2:StartWhen      rts:1015
rts:1015 ro:instanceof     c2:TimeEvent
rts:1015 c2:NoLaterThan    “110800ZAUG2025”
```

The first represents the assertion that the mission with identifier #1004 starts at time event #1015. The second represents the assertion that entity #1015 instantiates the TimeEvent universal from the C2-Test-Ontology, and the third represents the assertion that the time event #1015 occurs at a time point which is not later than the time specified by the “110800ZAUG2025” date-time stamp. The execution of rule red_7 shows (1) how redundant entities and their relations can be removed from military messages in such a way that (2) an ontology can be used to infer automatically assertions about the entities in the real world which are referred to in these messages.

Note that in this second step, the middleware executes only rules of types 1) and 2). Execution of deduction and reduction rules over the triple set whose visualization is shown in Fig. 2 then yields a new triple set whose visualization is shown in Fig. 3. This new triple set conforms to sound logical and ontological principles; specifically: there is here no redundancy, and all nodes and edges in the graph are ontologically well-formed.

Each node of the graph whose label is prefixed with ‘rts:’ contains the IUI of an entity referred to in the message, as follows:

- #1004 denotes a particular military mission
- #1015 denotes the particular TimeEvent when mission #1004 must start
- #1005 denotes the military unit that must perform mission #1004
- #1009 denotes the line along which the #1005 unit has to move
- #1014 and #1012 denote distinct points corresponding to the start and end of the #1009 line.

C. Step 3: Entity Tracking and Semantic Verification

In the third step, rules prefixed with ‘map_’ are executed over the triple set that results from step 2. The middleware first checks whether the RTS is empty. If it is, there cannot be any duplicate entries and the triple set is simply inserted into the RTS and processing stops. However, if the RTS is not empty, then the middleware program continues to execute this step to. Suppose the middleware receives a second military message (shown in Fig. 4) and that the RTS database contains the triples shown in Fig. 3. Here, the military unit named ‘2TF A TEAM’ is assigned the IUI rts:2005. However, in the RTS used in the message analysis system there is already a

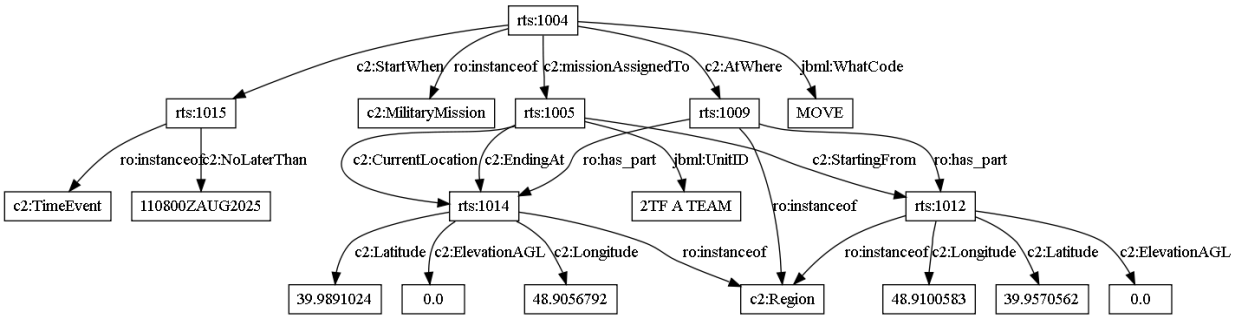

Fig. 3: Visualization of Ontologically Articulated Content of the XML message in Listing 1

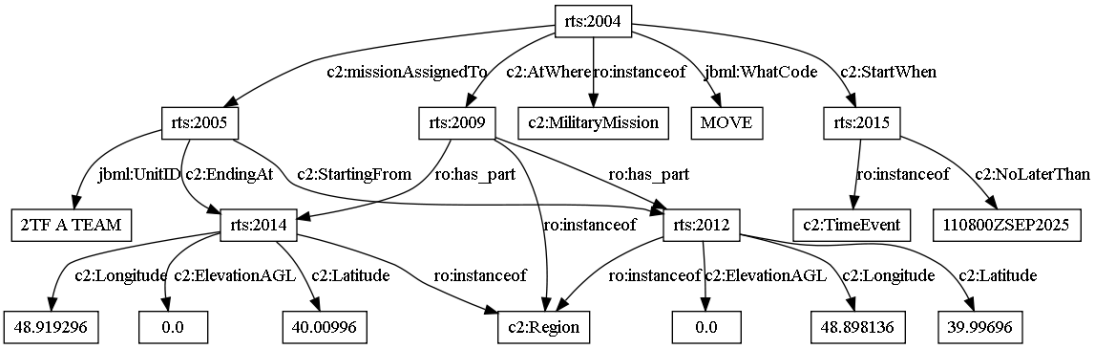

Fig. 4: Second Military Message after Processing via Step 2

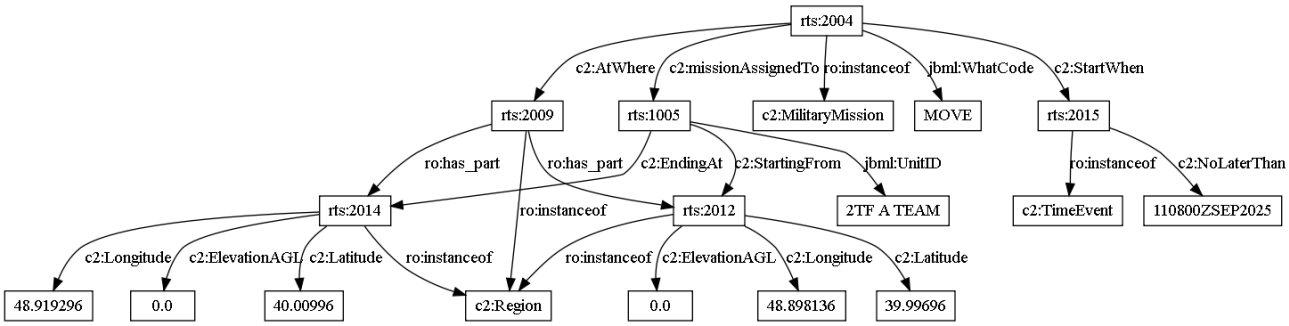

Fig. 5: Second Movement Mission as Described in the 2nd Example Message

military unit whose name is ‘2TF A TEAM’, to which the IUI rts:1005 has already been assigned.

The Referent Tracking principles state that if the military unit in the second message is the same as the one referred to already in the RTS, then it should continue to be denoted by the IUI rts:1005. It is this conformity to Referent Tracking principles that is achieved by the mapping rules.

A mapping rule whose conditions are satisfied by the incoming second message is:

[map_1: (?x jbml:UnitID ?a) -> entity_found(?x)]

This rule instructs the middleware to search for matching triples in the incoming message, in this case returning the triple:

rts:2005 jbml:UnitID“2TF A TEAM”

As ‘?x’ refers to an entity (whose IUI is to be determined), the middleware replaces the IUI in the triple found by ?x. The above triple is thus transformed into the following:

?xjbml:UnitID“2TF A TEAM”

The map_1 rule now instructs the middleware to search for a corresponding triple in the RTS which returns, in this case:

rts:1005 jbml:UnitID "2TF A TEAM"

As a result, rts:2005 is replaced by rts:1005 in all corresponding triples in the new message. The final output is shown in Fig. 5.

At this stage, all triples of the incoming message are in accordance with the RT principles, and the message triples are ready to be inserted into the RTS database in addition to the triple (rts:1005 jbml:UnitID "2TF A TEAM") which is already present.

At the end of these steps, an unambiguous representation of the data – at least in terms of the C2-Test-Ontology – is obtained, and additional reasoning over the data now becomes possible.

D. Step 4: Reasoning to Validate Incoming Messages

In this step, data validation rules are executed over the RTS database used by the middleware to check for any inconsistencies that may arise when inserting the content of the new message into the RTS.

Imagine a scenario under which the RTS is in the state shown in Fig. 3, and now the triples shown in Fig. 5, corresponding to a new planned mission for the 2TF A TEAM, are ready to be inserted. The new mission involves the movement of the team from location #2012. Because the new starting position is different from the current position of the unit as recorded in the database, it is not possible for the military unit to carry out the mission unless the starting point can be reached in due time. In this case, the middleware will provide an alert warning of potential inconsistency of the data.

To perform the reasoning necessary for such an alert to be generated, rules of type 4 are executed. The middleware executes these rules by communicating with the RTS and the reasoner. If the middleware finds any inconsistency as defined in a rule of type 4 (_val) then it generates warning messages.

One rule used in this reasoning scenario is:

```
val_1   (?a c2:missionAssignedTo ?b)
        (?a jbml:WhatCode ?c)
        (?b c2:CurrentLocation ?d)
        (?l jbml:missionAssigndTo ?b)
        (?l jbml:WhatCode ?c)
        (?b c2:StartingFrom ?o)
        notEqual(?a, ?l)
        notEqual(?d, ?o)
            ->          (?a    error:inconsistentWith ?l)
```

III. CONCLUSION

We are currently in a phase in which the middleware component is able to process several sample types of incoming messages and to perform a number of reasoning strategies in relation to incoming messages on the basis of information obtained through earlier messages. The benefit of using middleware along with rules built on the basis of a well-structured ontology such as our C2-Test-Ontology (itself an extension of UCore SL), is that, if changes occur, for example in the format of the military messages or in our understanding of military reality, changes will need to be made only in the rules and ontologies, and the need for further software changes will thus be reduced. Another benefit of this architecture is that the system can be re-used with other information systems pertaining to other Communities of Interest (CoI). The only changes we need to make are: (1) building an appropriate ontology for each new CoI, (2) writing a new parser to analyze the information coming from the corresponding information system, and (3) defining appropriate rules for that domain.

We also did not use rules that provide alternative interpretations of inconsistent data. Note for instance that the inconsistency as concerns the location of the 2TF A TEAM discussed above might result from a false assumption about which entity the name '2TF A TEAM' in fact denotes – either perhaps it denotes different units in different messages, or because in some messages typing errors have been made.

The approach can be used not only to reason with messages in given formats, but also to integrate messages with different formats. Future work should be based on fully axiomatized forms of the various ontologies. The C2-Test-Ontology should also be dramatically extended to include not only JBML, but relevant content from JC3IEDM, and NIEM with the objective of creating fully automatized interoperability corridors, establishing a model for further work not only on more comprehensive collections of messages, but also on information of other types.

REFERENCES

[1] J. Mark Pullen, Andreas Tolk, and C. Blais, "Joint Battle Management Language (JBML) -US Contribution to the C-BML PDG and NATO MSG-048 TA," in IEEE European Simulation Interoperability Workshop Genoa, IT, 2007.

[2] W. Ceusters and S. Manzoor, "How to track absolutely everything?," in Ontologies and Semantic Technologies for the Intelligence Community. Frontiers in Artificial Intelligence and Applications: IOS Press Amsterdam, 2009, (in press).

[3] S. Manzoor, W. Ceusters, and R. Rudnicki, "Implementation of a Referent Tracking System," International Journal of Healthcare Information Systems and Informatics, vol. 2, pp. 41-58, 2007.

[4] W. Ceusters and B. Smith. Strategies for Referent Tracking in Electronic Health Records. J Biomed Inform. 2006 Jun;39(3):362-78.

[5] The Universal Data Core, Army Net-Centric Data Strategy (ANCDS), http://data.army.mil/datastrategy_universal_core.html, (2009).

[6] K. Munn and B. Smith (eds.), Applied Ontology: An Introduction, Frankfurt/Lancaster: ontos, 2008, 342 pp.

[7] HP Labs Semantic Web Research, "Jena- A Semantic Web Framework for Java," 2009.

[8] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. L. Rector, and C. Rosse, "Relations in biomedical ontologies," Genome Biology, vol. 6, p. R46, 2005.