Exploring Primitives of Meaning in Support of Interoperability

Charles D. Turnitsa
Virginia Modeling Analysis and Simulation Center
Old Dominion University
Suffolk, VA 23435
cturnits@odu.edu

Andreas Tolk, Ph.D.
Frank Batten College of Engineering & Technology
Old Dominion University
Norfolk, VA 23529
atolk@odu.edu

Robert Kewley
Operations Research Center
US Military Academy
West Point, NY 10996
Robert.Kewley@usma.edu

Keywords:

Primitives of Meaning, MATREX FOM, Command and Control Languages, Semantic Alignment, Conceptual Modeling

ABSTRACT: *Semantic mismatch between systems is due, in part, to the grouping together of terms who have defined meaning in different levels of granularity, and which are composed together into different groupings by distinct systems. It has been proposed that making use of elemental concepts (referred to here as primitives of meaning) can assist in interoperability, but seeking to define all terms at a level of granularity equal to or greater than that of all involved systems.*

By decomposing a system's groups of composed terms into primitives of meaning, the building blocks that can be reassembled into the compositions required by another group (of another system, for instance) can be made apparent. While such a de-composition could serve as the basis for an interoperability enabler, having the decomposition available as a common descriptor to highlight areas of semantic misalignment should prove in itself useful.

Taking doctrinal statements for US small unit infantry actions as one semantic system, we show how the elemental ideas that are grouped together into commands can be identified and isolated for reconstruction into other groupings. This is the first research step towards relying on primitives of meaning for interoperability.

1 Introduction

This paper presents research that is intended to contribute to interoperability solutions by showing how the primitives of meaning that make up the terms (and, in turn, the compositions of terms) that are the basis of communication between systems attempting to interoperate. A number of publications from both inside [1] and outside [2] of SISO have previously described the Levels of Conceptual Interoperability Model, which gives stratification to the continuum of potential conceptual meaning available to systems attempting to interoperate. By its nature, a system that uses atomic conceptual elements (such as the primitives of meaning, described here) in order to exchange semantic meaning between systems is in support of the conceptual interoperability at or above level 3, the "Semantic Level" of interoperability.

The paper is organized in the following manner: section 2 describes the problem of semantic misalignment between systems, such that the rest of the paper can be viewed with clarity as supporting a specific problem. Section 3 describes a possible solution for the problem in general terms. Section 4 describes a particular problem and applies the general terms of the solution to that problem to illustrate how it might be practically applicable. Section 5 describes some other works that explored similar ideas. Section 6 gives a summary, and describes the way forward for future research.

2 Semantic Misalignment

Interoperability between systems is enabled by the transmission of communications from a transmitting system to a receiving system, and the interpretation of those communications by the receiving system. By using this mechanism, one system can make information about its internal state known to another system. Such communications have different names, but in the community of modeling and simulation (M&S), especially military M&S, these are known as "messages" or "updates". When the communications are intended to convey command and control (C2) information, they are often thought of as "orders", "reports", "tasks", or "plans". In almost all cases, such communications represent a script of several actions, or

represent one or more actions for an aggregation of one or more objects.

While such groupings are convenient, and avoid having to send multiple communications to represent a common idea, between different systems the grouping is possibly arranged at a different granularity or composition. This problem is represented graphically in Figure 1- Misalignment of Compositions.

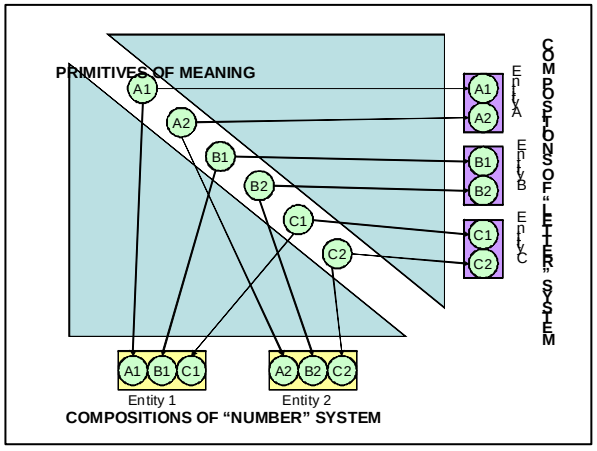

Figure 1- Misalignment of Compositions

In Figure 1, we can see that two different systems – named letter system and number system – have different compositions. They may be discussing the same letter-number items (a1, a2, b1, b2, etc.), but they compose them along different axes of categorization, based on their own perspective.

When these compositions are groupings of actions, or processes, then each composition can be said to represent a script of processes. The likely problem here is that for one system, a series of processes may always be encountered in a specific script, but for another system each primitive (or component process) of the script may need to be represented separately, or combined in scripts of different composition. Likewise, when the composition is referring to an aggregation of different objects which is convenient for one system, which we can name a collective of objects, it is quite possible that another system may have to address only one member of such a collective, or perhaps a different collective composed of other primitives.

3 Primitives of Meaning

The proposed solution, of breaking down the compositions of each system, into higher granularity primitives that can be reassembled at need for each system in question, is a solution that works very well when dealing in the abstract, but raises some immediate questions when it is brought to the applications table.

The immediate question asked when attempting to break a composition into primitives is concerned with identifying the primitives themselves. Simply put, a primitive is some identifiable conceptual aspect of a composition. When one is considering an entity, apart from how it might be represented as a composition within a system, it is clear that there might be an unknowable number of different concepts/aspects that make up the definition of that entity. But, as a composition of attributes represented in a system, the entity has (at most) one primitive per attribute (although it likely has many fewer than this, so that each primitive is responsible for several attributes).

If one is attempting to identify the primitives that make up an action, or process, composition, it can also be broken up based on its identifiable component parts. As the action or process is responsible for altering the attributes of some other entity (either object or process), then the different ways in which those attributes can be altered can be identified separately as primitives. For example, suppose that a process composition is actually a script of several actions that each modify an object's attributes, in a time sequential manner, than each identifiable action that makes up part of the script could be labeled a separate primitive.

3.1 Reasons for identifying primitives of meaning

Once the several primitives of a composition are identified, the next question is likely to be – how is my system going to handle this collection of primitives, when it previously handled one or more compositions in their place. This question is likely to have a number of different answers; two that have been anticipated are described here.

- *Case 1*: Primitives are identified and relied upon for building a transport/translation mechanism between systems.
- *Case 2*: Primitives are identified as having a requirement for representation within a system, which is independent from the entities that originally exhibited them.

These two are distinguished from each other based on why the primitives are used. In the first case, if the primitives are used merely as a transport mechanism, when the different compositions are out of alignment with each other, then the solution becomes one of decomposing and recomposing for each system. Going back to the example of Figure 1, if letter system requires the entity "a1a2", and to get that information the number system must provide the compositions "a1b1c1" and "a2b2c2", then a translation mechanism

must be able to decompose and recompose from compositions to primitives, and so on, for each system. The original systems remain intact.

The second case is one where it is required to separate out the primitives out of an entity for distinct consideration within a system. For instance, if a system is used to executing a process composition, which is a script of different actions, and it now becomes required for the system to distinguish a subset of that composition – distinguishing one action, or a smaller composition of actions – then the system must be modified to only act on the desirable portion of the original composition. While not possible in all situations, a potential approach case 2 involves having the translation mechanism apply null or zero attribution to the portions of a composition that are not desired to be represented in the system. It is also quite possible that a combination of these two cases will be required to solve the applications problem at hand – this would be in the case where the target system is now required to exhibit portions of a new composition, where it used to show a different composition. In order to derive the portions of the new composition that is required, a translation (as in case 1) must be accomplished.

3.2 Primitives of meaning and MBDE

Model Based Data Engineering (MBDE) is a useful method for structuring data for purposes of enabling interoperability, based on the intended model that the system relies on for giving the data meaning. MBDE consists of following four steps [3], which are Data Administration, Data Management, Data Alignment, and Data Transformation. In non-trivial cases, it is likely that the different model perspectives of separate systems will require a case 1 consideration of breaking entities into their component attributes. This means that the four steps of MBDE can be assisted by considering the primitives of meaning that are responsible for those attributes, for the purposes of organization and categorization of the attributes to be handled during Data Alignment, and Data Transformation.

3.3 Primitives of meaning and the LCIM

The Levels of Conceptual Interoperability Model (LCIM) is a model for stratifying the possible continuum of conceptual meaning that can be expressed between interoperating systems. It is divided up into a number of different layers. The semantic layer relies on expressing information between systems, and giving it a semantic label (a name or tag assigned by the originating system that hopefully is within the context of the receiving system). To move above that level, to the increased conceptual expressivity described for the pragmatic level and beyond, the meaning and context of the information exchanged must be made understood by the originating system to the receiving system. In a community where the primitives of meaning are unambiguous, and for systems whose entities can described (using a method such as described in case 1, from section 3.1, above) using those primitives of meaning, then the meaning of the information exchanged can be made known from system to system.

4 Doctrine to FOM

In an ongoing project in support of PEO Soldier, it became desirable to depict doctrinally based tasks, for small infantry units, in a number of different simulation systems. Those systems are federated together using the high level architecture (HLA), and relying on the MATREX federation object model (FOM). The FOM gives description and attribution to a number of entities and interactions (actions) that are appropriate for small unit actions. However, some of the interactions that are typically relied on for the tasks in question have a lower composed granularity of meaning, than would be desired in a training setting. This seemed like a very good test case to attempt to solve the problem using primitives of meaning.

4.1 Doctrinal Model

The desired action to depict for a typical training exercise that this experimental work focused on, is that of representing a “support by fire” order within a constructive simulation, with appropriate input from members of the training audience to represent the input that a small unit commander would give to his unit.

The support by fire command is described in US Army Infantry doctrine [6]. The description of the command is given in the appendix (section 7) of this paper. From the field manual, we can take a look at an example order that a unit leader might receive, that describes how the actions of a support by fire command would be transferred, in a C2 community.

> *WPNs, establish an SBF on the berm to the south of Bldg 100 and orient your fires to the North. On the codeword “ANVIL 1,” engage the first and second floor with 10 secs of cyclical M240B fire, subsequently talking your guns at a rapid rate. Additionally, maintain a watch and shoot to the East of Bldg 100 to prevent the enemy’s exfil. On “HAMMER,” shift your fires to the east of PL RED onto Bldg 101, and standby for “ANVIL 2.”*

Combining the knowledge of the general form of the doctrinal command, with a specific example, we can now take a look at the command, and begin to see which processes/actions are required to be depicted, and what their features are; likewise, we are able to get

a catalog of the objects that need to be represented in order to satisfy those processes.

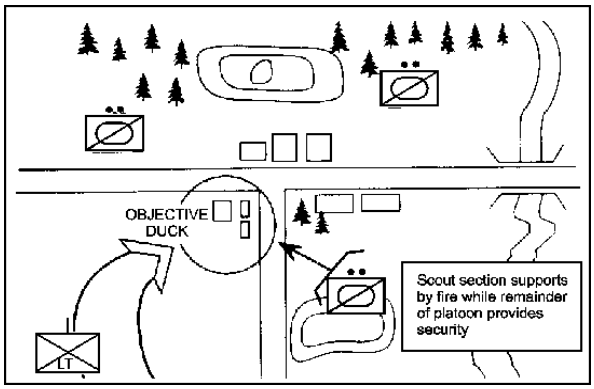

Figure 2- Diagram supporting example of Support by Fire

The basic processes involved are these:

- Move to given location of SBF (Support by Fire), and take up position, using given Orientation
- On "engage" code word, fire at stated target
- On "shift" code word, cease firing, re-orient, and begin firing at new stated target
- On "lift" code word, cease firing

If we begin to categorize these processes, along with the objects that are needed within the model to enable them, we come up with:

- UNIT will MOVE to LOCATION
- UNIT will ORIENT in a stated direction, once at LOCATION
- UNIT will await "fire code word"
- UNIT will await additional "code words"
- On "fire code word", UNIT will begin to FIRE at stated TARGET
- On "lift code word", UNIT will cease the FIRE activity
- On "shift code word", UNIT will cease the FIRE activity at the current TARGET, re-ORIENT, and begin to FIRE at a new stated TARGET

4.2 Object/Process Taxonomy

We begin to see, in our process of examining the doctrinal model, to derive a taxonomical structure of the objects and processes required to simulate (or depict) this model. By dividing these up into objects and processes (separate lists), we can see the following begin to emerge:

Objects

Unit – the military unit being ordered to assume the SBF. UNIT is an explicit object.

Location – the spot on the map/chart where the SBF is to take place. Location is an explicit object.

Target – the location where the fire is to be directed at. Target is an explicit object.

Map/Chart – an implicit object, represented by a reference system that provides a domain and range for the locating objects and describing actions.

Class V Expendables – if these are modeled by the simulation, then there will be an expenditure rate that depletes these during the existence of a FIRE process. If the expendables are depleted, then the FIRE process implicitly stops.

Processes

Move – the UNIT changes its location on the map/chart, to match the LOCATION given in the order. MOVE is a process that will end as soon as the UNIT's LOCATION is the same as given with the MOVE process.

Orient – the UNIT changes it's "facing" to match the one given with the ORIENT process command. ORIENT is a process that will end as soon as the UNIT's orientation matches that given with the ORIENT process.

Fire – the UNIT begins to discharge its weapons at the stated TARGET. The FIRE process is continuous, so a start and stop for the period of fire is indicated by the corresponding code words

In addition, each of these processes needs to have a subject start/stop condition, based on the command words stated by the unit commander.

4.3 Granularity of Meaning

From one model (the model of US Army Infantry Doctrine) we see the level of granularity that we are required to depict. The other model we have is the MATREX FOM based HLA federation.

Within the MATREX FOM, appropriate entities exist that can show all of the objects, with the appropriate level of attribution, that we have identified from our Doctrine model. We have Units, Locations, Targets, a Map object, and (when necessary) Class V expendables – all modeled at the level of depiction required by the doctrine model.

Moving from objects to processes, however, we have a problem with the MATREX FOM. The interactions that depict movement, orientation, firing, etc are often grouped together, and the constructive simulation that receives direction to depict that interaction does so using its own interpretation. This removes the need/ability to have the member of the training audience be the one to initiate the action using a

simulation of issuing the command code words (the fire code word, the lift fire code word, the shift code word, etc.). But by removing that need, it also removes the training possibility of having the audience member issue commands.

4.3.1 Object Granularity

As mentioned, the objects of the FOM remain similar to the identified objects of the Doctrinal model. Here is a short synopsis of the FOM objects and parameters that suffice.

- **Unit** – The FOM object *AggregateUnit* is a candidate here. It gives parameters for all of the information concerning a unit's current status (ID, type, side, country, damage, ammo, fuel, appearance, location, orientation, etc.)
- **Location** – The FOM element *LatLongAltPositionCDT* is a complex data type (the suffix CDT is an indicator), that is part of the *StateVectorCDT*, which is the *UnitLocation* parameter's data type (*UnitLocation* is from *AggregateUnit*, and other objects). This CDT gives the x,y,z location of the object it is with. As mentioned, this is part of the StateVectorCDT which also gives velocity, position, orientation, orientation rate, orientation acceleration, as well as a timestamp. Changes to the StateVectorCDT represent the results of a **Move** process.
- **Target** – This is also based on the *LatLongAltPositionCDT*, which is given as a parameter to the *AttackByFire* interaction (the parameter is named *EngagementArea*). This complex data type, when used for this parameter, is given with a cardinality of 2+, as it is intended to be the vertices of a region (opposite corners of a quadrilateral).
- **Map** – While not given specifically as an object, or a set of parameters, the **Map** gains definition from the boundaries it sets on all the uses of *LatLongAltPositionCDT*. This boundary is the results of an assumption, that all positions will be within the bounded region of the **Map**.
- **Class V Expendables** – One of the parameters of the *AggregateUnit* object has the complex data type *AmmoStateCDT* as its possible value – this is the means for aggregate units to track their own class V (ammunition) supplies in the FOM.

As mentioned, these objects are fine, in terms of matching granularity from the Doctrinal model to the MATREX object model (FOM). It should be pointed out, in the examples given above, that we chose one of two possible "cells" of objects from the MATREX FOM. The FOM supports a number of objects and interactions at the entity level, and another set of objects and interactions at the aggregated unit level. For simplicity, these examples are from the second cell, the aggregated unit level. The recommendations made below (in section 5), however, are for both cells.

4.3.2 Process Granularity

Now, to take a look at the processes as they are represented in the FOM (by HLA interactions). These are intended to proved the interactions to represent those required by the Support by Fire exercise, from the Doctrinal model.

- **Move** – The interaction from the FOM is named *Move* (not surprisingly). The *Move* interaction has, as parameters, the *Destination* of the move (which is a *LatLongAltPositionCDT*), the *Route* followed (which uses the *SimpleWayPointCDT*), and a rate of speed, which is given in meters per second.
- **Orient** – This process is accomplished via the transmission of *LatLongAltPositionCDT*. This is given as part of *Move*, and is a parameter of that interaction. In order to change the current orientation, issue a *Move* interaction, with the current location as the destination, but with a new orientation.
- **Fire** – Accomplished within the aggregate unit "cell" by the interaction *UnitAttack*. This interaction gives the unit information (via CDT parameters) on where to move to, and where to attack to. It includes all three of the actions given here, Move, Orient, and Fire, into one interaction, with assumptions about the ordering and the concurrency of the action.

As can be seen from these actions, there are some assumptions made in the granularity of activity as it is presented within the FOM. The actions are grouped together, with assumptions made about the concurrency of the actions, or whether or not they should always go together. It would be possible to obviate one or more of these granular pieces within the interactions (by, for instance, giving the current unit's location to Move, but giving a different Orientation parameter, thus accomplishing an Orient, without moving at all), but this is making assumptions about how the actions will be ordered, and leaves out the possibilities given earlier for training audience participation. In addition, when a number of distinct processes are scripted together (as in our example case of Support by Fire), but are separated by other user-initiated C2 commands (or any other interruption in a scripted series of processes), then it is not possible to identify a future interjected process as being part of the initial script, or something new.

Based on this analysis, it appears as if the granularity of the FOM is at a lower level of process fidelity than is required from the Doctrinal model. Because of this, some recommended additions to the MATREX FOM have been identified, organized according to the principles of Primitives of Meaning for processes (distinct, atomic actions, not aggregated together).

5 Recommended FOM Additions

In order to fully support the required granularity of the Doctrinal model, some of the functions of a command that are represented by the current interactions of the MATREX FOM need to be identified and isolated. Most of these have to do with separating the C2 functionality from the command. This is because of the specific example used – Support by Fire. In representing that task, the Doctrinal model specifies that the command (SBF) is given, then elements of the command are performed following command "trigger" words.

5.1 Process Modeling – Activity vs. Results

Before proceeding with the identified new FOM interactions, it is worthwhile here to discuss a topic that is at the heart of modeling decisions, but is often overlooked. This topic has to do with the modeling of processes. Note that at this time, the discussion here is specifically talking about modeling (which is making the decisions about what to represent from the referent), and not about simulation (which is the implementation of the decided-upon model). When modeling a process, two approaches can be taken. Modeling the activity, or modeling the results. The difference is highly subjective in that given different perspectives, one can easily become the other, but within the same perspective.

When modeling activity, one actually decides to depict what the changes are that the objects affected by the process are going through, at the highest granularity of time permitted by perspective of the modeler. For instance, if it is desired to model the movement of an infantry platoon, moving dismounted at 1 kilometer every 15 minutes, in a model that will have a time granularity of 1 time tick per minute of represented time, then over the kilometer of modeled terrain, the activity based model would show 15 different positions along the way, one for every time tick.

Using the same example, when modeling the results of the process, even though the model might have a time granularity of 1 minute time ticks, the modeled infantry platoon might be designated as being "in movement", and only have its destination updated in the model once it is reached (15 modeled time ticks later).

There are good, practical reasons for taking the second approach (that usually have to do with the simulation of the model, not the model itself), but it should be realized by the modeler that he is sacrificing some possibly important details if he makes the decision to model results rather than activity. In our case, the FOM interactions model the results, rather than the activity. The individual simulations that get those interactions, if they want to depict the activity, must make their own assumptions, as the model does not describe them.

Once we arrive at a process modeling specification (currently being researched at VMASC), it can be evaluated and determined if the benefits of modeling activity at a high resolution leads to better simulation (of activity OR results). A new way of thinking about modeling is to describe the process in a manner that is true to the referent as possible, so that in the example given above, dividing the move up into time slices would not have to be done at all in the model, especially if the process (movement) remained the same throughout the period of interest. In that case, it could say that movement from point 1 to point 2 takes place (leading to a continual change in location with respect to time, but not discretized to time slices), without worrying about how to devolve it to the system that will implement the model. This is a true implementation-neutral approach to modeling processes.

5.2 FOM Additions

It is proposed, to support the other FOM additions being described here, that there be a new base interaction added, that other interactions can extend. This same base interaction will be used as the basis for extension for both the entity level cell and the aggregated unit level cell. The new base interaction is called *C2Command*, which is similar to the existing base interaction *Command*. *C2Command* will extend the *Communication* base interaction. It has the following parameters:

- *ContributingID* has a cardinality of 0+, based on the *FederateIDCDT*, which comes from the *SituationReports*, used in the same way that the interaction *Command* uses this parameter. It is for identifying the federation object that is issuing the command.
- *StartSignalID* has a cardinality of 1, and a data type of long integer. It correlates to the *CommandID* parameter that will be used to start an extended interaction based on *C2Command*. This is the C2 start signal identifier.
- *StopSignalID* has a cardinality of 0-1, and a data type of long integer. Correlates to a CommandID to stop the C2Command. Only needed for

interactions that extend this command who require a "stop" command - like *C2Fire*.

Each new proposed interaction introduce here, that extends *C2Command*, will therefore be given a start signal identifier, but not necessarily (except where appropriate) a stop signal identifier.

5.2.1 Entity Level Interactions

Based on that new base interaction, here is a list of the interactions designed to work with the entity level objects.

- *C2Move* - (similar to *Move*, could also be based on *MoveAlongRoute*) Extends *C2Command*. It has the following parameters -
  - *Destination* (has a cardinality of 1 and a data type of *LatLongAltPositionCDT*). This gives the destination point, within the bounded parameter values, of the Map object.
  - *Route* (has a cardinality of 1+, and a data type of *SimpleWayPointCDT*). This gives the points to follow along a simple way point route, for the movement.
  - *Rate of Speed* (has a cardinality of 1 and a data type of long integer). This is in kilometers per hour.
- *C2Orient* - (new interaction) Extends *C2Command*, and has the new parameter *Orientation* (a cardinality of 1, with a data type of double float) this is the new orientation of the entity, given in degrees from due north, increasing clockwise.
- *C2Fire* - (similar to *Assault*) Extends *C2Command*. It has the following parameter, in addition to those inherited from *C2Command*, *EngagementTRP* (a cardinality of 2+, with a data type of *LatLongAltPositionCDT*) Gives the engagement area, by specifying the vertices of a quadrilateral.
- *C2CommandSignal* - (new interaction) Extends *Communication*. It has the parameter *CommandID* (a cardinality of 1, with a data type of long integer) Supplies the *CommandID* which will be either a start or stop signal for one of the other interactions.

5.2.2 Unit Level Interactions

Following is a list of the aggregated unit level interactions that are recommended. Note that these are very similar to the entity level interactions, with the addition of parameters needed to describe the behaviors of an aggregated unit performing the same activity.

- *C2UnitMove* – similar to *C2Move*, with the additional parameter of *formation*, which has a cardinality of 1 and uses the *MoveFormationEDT* data type.
- *C2UnitOrient* – similar to *C2Orient*, with the additional parameter of *OccupyPosition*, which has a cardinality of 1+, and a data type of *LatLongAltPosition*. The purpose for the new parameter is the give the occupational space of the unit once it adopts its new orientation.
- C2UnitFire similar to C2Fire, with the same parameters.
- C2UnitCommandSignal – Similar to C2Command, with the same parameters.

5.3 Intended Use

The basic use of these interactions would be, when an order is issued to a unit, to send one of the C2 command interactions (either C2Move, C2Orient, or C2Fire), which will have as its parameters, identified command codes for starting, and (in the case of C2Fire) stopping the activity. Such an interaction would be received by the federate, but not acted upon until a later interaction (C2CommandSignal) was transmitted to it, with a parameter having an issued command code that signifies the earlier interaction to now begin. In this way, the transmission of orders, and the transmission of commands by a member of the training audience can be simulated. An alternative to having he start command code supplied with the C2 command interaction, would be a default behavior of "start now" if no code were given. The problem here, is one that was identified within the literature of the existing FOM interactions – and that is there is no "suspend fire" or "halt fire" interaction that has been identified. Perhaps this is interpreted by the individual federate, or perhaps there is some workaround that experts with the FOM are accustomed to using.

6 Primitives of Meaning and Combat Modeling

The example given here is only in support of the analysis of one command, from US Army Infantry doctrine (the Support by Fire task). If a robust decomposition of the US Army Doctrinal model were to be performed, then a number of additional primitives of meaning might be identified, other than just move, orient, fire, and command. However, the number of primitives, it is assumed, would be much less than the number of tasks and command that could be supported by their composition into tasks. A benefit would be the achievement of a desired granularity of meaning, for the purposes of depiction of the model, and also the means to convey meaning of semantic terms (supporting level 4, or Pragmatic, expressivity between systems, according to the LCIM – see section 3.3 above).

6.1 Combat Object Primitive Capabilities

In fact, there has been some work identifying some of the primitives of meaning for processes that are performed, at least in the kinetic realm, of battlespace activities. One such work is [4], which describes the modeling of battlespace objects in terms of their basic capabilities. The objects are based on agglomerating these primitive capabilities onto a basic conceptual entity called an "object for military operations". The identified capabilities are:

- Attrition – neutralizing other objects / preventing neutralization of own objects
- Movement – changing locations / preventing others from changing locations
- Transportation – transport other objects
- ISR – detect and identify other objects / prevent detection
- Communication – transport data and information / jam communication
- C2 – decision cycle to generate situation adequate commands
- Maintenance – repairing objects
- Material supply – use of material in the operation

The first six of these are the basic combat activities, and the last two are considered combat support activities, according to the original report.

Parameterization of the resulting entity would give the details. Presumably, an entity that was composed (in part) of the Movement capability, would then be capable of being affected by Movement processes, depending on the simulator in question that implemented such a model.

6.2 Decomposed Role-Behaviors

Another more recent work [5] shows how Joint Mission Threads can be used to illustrate how military units are tied to their role-behaviors (the processes that they are expected to perform in satisfying a mission thread, or in response to battlefield conditions). These role-behaviors are identified as having several C4I behaviors, and several tactical behaviors. A list follows.

- Update Situational Awareness (C4I)
- Informational Message (C4I)
- Call for fire Message (C4I)
- Other Message (C4I)
- Command Self (tactical)
- Command Subordinate (tactical)
- Maneuver (tactical)
- Fire/Engage (tactical)
- Change Status (tactical)

Comparing these role-behaviors to the primitive capabilities discussed earlier in this section reveals some overlap. In the second listing, the command and control activity is given some additional granularity, by breaking it up into a number of different C4I/Message type activities, but the other actions correspond pretty well. That these two divergent approaches have identified very similar primitives indicates that there may be some community level uniformity here.

6.3 Primitives of Meaning and Modeled Primitives

So far, in this paper, the authors have discussed decomposing model entities (objects and processes) into primitives of meaning that have implemented attribution or parameters. In fact this represents a limited, but very much practical, subset of the idea of primitives of meaning. What is presented here could be called modeled primitives, and in addition to them, there also might exist within a system a number of primitives of meaning that influence the relationships and structure of the system (therefore having an influence on attempts to interoperate with such as system at levels 5 and 6, Pragmatic and Conceptual, of the LCIM), as well as its constraints and assumptions. While the difference between the two groups (Primitives of Meaning, and Modeled Primitives) is important, and the existence of each can be shown, it is beyond the scope of this paper to explore this relationship. A future paper will no doubt explore this paradigm, but for now it is enough to identify the difference.

7 Summary

Primitives of meaning remains a topic to be fully explored, however this early analysis/application indicates promising results. The idea is to express the meaning of an object or process within a model by identifying its atomic elements of meaning, at or above a granularity required by the highest fidelity use they will be put to. As each of these primitives will then suggest one or more of the attributes that combine to define the original object or process, those attributes can be addressed by other objects/processes in the normal interaction that a dynamic model provides.

The US Army Infantry action, Support by Fire, was examined and analyzed to determine the atomic processes and objects that must be depicted in order to accurately model such a task for a simulation system to illustrate those objects and processes for training. Another model, one that is currently being used by PEO Soldier to depict tasks to simulation systems for

training, the MATREX FOM, was also analyzed. It was shown to be at a level of granularity, in terms of the processes required for support by fire, too high to provide for proper training objectives. Therefore new FOM interactions have been identified that can depict the processes at a primitives of meaning level (i.e. – each process is one atomic activity).

Future work remains to have systems that can use the new FOM interactions, and to test them to see if the primitives (and their composition) does provide the results anticipated. Additionally, further analysis of other doctrinal tasks and commands awaits to identify further primitives (both of objects and processes) to assemble a new series of FOM objects and interactions at the primitives level.

Additionally, future work awaits the development of (1) a process modeling specification (currently being researched at VMASC), in order that engineering methods can be applied to the composability and interoperability of the processes of models, and (2) further exploration of the primitives of meaning paradigm in modeling, and additionally, the difference of primitives of meaning from modeled primitives.

Acknowledgments

The underlying research was partly supported by the PEO Soldier supported interoperability work, designed to explore the federating of several systems using the MATREX FOM. Additional ideas contributing to the research presented here came from research performed in support of the development of the Coalition Battle Management interoperability standard, SISO PDG-CBML. Finally, the idea of primitives of meaning arose out of ongoing PhD research performed by the primary author (Turnitsa), and his chief advisor (Tolk).

8 Works Cited

[1] *Data, Models, Federations, Common Reference Models, and Model Theory.* Tolk, Andreas, Diallo, Saikou Y and Turnitsa, Charles D. Genoa : IEEE, 2007. Proceedings of the 2007 European Simulation Interoperability Workshop. pp. 07E-SIW-052.

[2] Tolk, Andreas, Turnitsa, Charles and Diallo, Saikou. Implied Ontological Representation within the Levels of Conceptual Interoperability Model. *International Journal for Intelligent Design Technologies.* 2008, Vol. 2, 1.

[3] Tolk, Andreas and Diallo, Saikou. Model Based Data Engineering for Web Services. *IEEE Internet Computing.* 2005, July.

[4] IABG Report B-CS 2027/02: "Entscheidungsunterstützende Systeme (EUS)" [in German; Decision Support Systems], Ottobrunn, Germany, July 1999.

[5] Works, Paul. "M&S Decision/Role-Behavior Decompositions," Wargaming and Analysis Workshop, Military Operations Research Society, October 2007.

[6] US Army Infantry School, "The SBCT Infantry Rifle Company," Department of the Army, Washington DC, January 2003.

Authors' Biographies

CHARLES TURNITSA is a Senior Project Scientist at the Virginia Modeling Analysis and Simulation Center at Old Dominion University. In addition he is also a Ph.D. Candidate, studying under Dr. Andreas Tolk at ODU. He has a M.S. in Electrical and Computer Engineering from that institution.

ANDREAS TOLK is Associate Professor in the Faculty for Modeling, Simulation, and Visualization at the Engineering Management Department of the College of Engineering and Technology at Old Dominion University (ODU) of Norfolk, Virginia. He is affiliated with the Virginia Modeling Analysis & Simulation Center (VMASC). His domain of expertise is the integration of M&S functionality into real world applications based on open standards. He received a Ph.D. and an M.S in Computer Science from the University of the Federal Armed Forces in Munich, Germany.

ROBERT H. KEWLEY is currently the Director of the Operations Research Center at the United States Military Academy (USMA) Department of Systems Engineering. He was commissioned in 1988 from the USMA as an armor officer. His armor assignments include Task Force 1-32 Armor in the 1st Cavalry Division and Task Force 1-70 Armor in the 194th Separate Armored Brigade. He served as a tank company executive officer during Operations Desert Shield and Desert Storm. He has commanded both a tank company and a cavalry scout training troop. His analysis experience includes a teaching assignment at West Point and a tour at the Center for Army Analysis. LTC Kewley's research interests focus on command and control systems. He has a M.S. in Industrial Engineering and a Ph.D. in Decision Science and Engineering Systems, both from Rensselaer Polytechnic Institute.

9 Appendix

This is the text of the doctrinal definition of the Support by Fire command that is the subject of the word described in this paper.

SUPPORT BY FIRE

The platoon maneuvers to a position on the battlefield from which it can observe the enemy and engage him with direct and indirect fires. The purpose of support by fire is to prevent the enemy from engaging friendly forces.

a. To accomplish this task, the platoon must maintain orientation both on the enemy force and on the friendly maneuver force it is supporting. The platoon leader should plan and prepare by:

- Conducting line-of-sight analysis to identify the most advantageous support-by-fire positions.
- Conducting planning and integration for direct and indirect fires.
- Determining triggers for lifting, shifting, or ceasing direct and indirect fires.
- Planning and rehearsing actions on contact.
- Planning for large Class V expenditures, especially for the weapons squad and support elements since they must calculate rounds per minute. (The platoon leader and weapons squad leader must consider a number of factors in assessing Class V requirements, to include the desired effects of platoon fires; the time required for suppressing the enemy; and the composition, disposition, and strength of the enemy force.)

b. A comprehensive understanding of the battlefield and enemy and friendly disposition is a crucial factor in all support-by-fire operations. The platoon leader uses all available intelligence and information resources to stay abreast of events on the battlefield. Additional considerations may apply. The platoon may have to execute an attack to secure the terrain from which it will conduct the support by fire. The initial support-by-fire position may not afford adequate security or may not allow the platoon to achieve its intended purpose. This could force the platoon to reposition to maintain the desired weapons effects on the enemy. The platoon leader must ensure the platoon adheres to these guidelines:

- Maintain communications with the moving force.
- Be prepared to support the moving force with both direct and indirect fires.
- Be ready to lift, shift, or cease fires when masked by the moving force.
- Scan the area of operations and prepare to acquire and destroy any enemy element that threatens the moving force.
- Maintain 360-degree security.
- Use ICVs and Javelins to destroy any exposed enemy vehicles.
- Employ squads to lay a base of sustained fire to keep the enemy fixed or suppressed in his fighting positions.
- Prevent the enemy from employing accurate direct fires against the protected force.