Ontology and Rule Based Knowledge Representation for Situation Management and Decision Support

Neelakantan Kartha, Aaron Novstrup[1]
Stottler-Henke Associates Inc, 1107 NE 45th Street, Suite 310, Seattle, WA 98105

ABSTRACT

Supporting human decision making in tasks such as disaster planning and time-sensitive targeting is challenging because of the breadth and depth of knowledge that goes into the decision making process and the need to reason with this knowledge within tight time constraints. Ontologies are well suited for representing the concepts that humans use in describing the domain of interest. However, ontologies can be costly to develop and ,by themselves, are inadequate to capture the kinds of decision making knowledge that arise in practice—for instance, those that refer to multiple ontologies or to established precedent. Such decision making knowledge can be represented by using a knowledge representation formalism that we call *decision rules*. These decision rules are similar to the rules used in rule based systems but can (a) include primitives from multiple ontologies and primitives that are defined by algorithms that run outside of the rule framework (b) be time dependent and (c) incorporate default assumptions. We report on our ongoing experience in using such a combination of ontologies and decision rules in building a decision support application for time sensitive targeting.

Keywords: decision support, ontologies, rule-based systems, time sensitive targeting

1. INTRODUCTION

Rapid and high quality decision-making that takes into account a large number of simultaneous constraints is a critical aspect of success in highly dynamic and complex environments such as the modern battlefield and the mission control of space vehicles. While we expect that expert human oversight will be an essential component in such complex decision-making for at least the near future, alleviating the cognitive burden on the human decision maker is critical for increasing the reliability of and decreasing the time for decision making. Technologies that aid human decision making are starting to make their presence felt and will become even more important in the future as the amount of information available for decision making continues to increase, whereas the time available for making critical decisions continues to be scarce.

In this paper, we report on our ongoing experience in building a decision support application (called *Sentinel*) for the problem of time-sensitive targeting. Time sensitive targets (TSTs) are those that present an exceptional opportunity or an imminent threat, and on which an attack can be carried out only in a limited time window. Currently, the decision for targeting and engaging a TST is done, for the most part, by a TST cell, which consists of offensive-duty officers, intelligence officers and administrative support. The TST cell is responsible for determining valid TSTs and for determining the appropriate course of action (COA). *Sentinel* helps the users in determining the appropriate COA, taking into account different aspects, such as whether the target is withheld or on a no-strike list, what the expected collateral damage is and what the legal ramifications are for pursuing the target into enemy territory.

The desired capabilities of *Sentinel* can be summarized as follows:

- Integrate different considerations (such as the rules of engagement, considerations about collateral damage, etc.) to decide whether the TST should be pursued and determine if the cost of the engagement is acceptable given the Joint Forces Commander (JFC) objectives and current situation in the battlespace.
- Determine the set of available resources capable of engaging the TST, and the impact on existing missions and on the JFC operational objectives when a resource is retasked to engage the TST.

[1] [kartha,anovstrup]@stottlerhenke.com

- Present this information to distributed users without overwhelming them with details, while at the same time allowing them to drill down for details, if needed.
- Present analyses that will assist the decision maker in selecting an option. This will include using metrics such as impact on other missions, likelihood of success and likelihood of the assigned asset surviving the strike, as well as imposing a ranking on the options based on this analysis.
- Allow the user to easily suggest an alternate COA and provide the same analysis for the user's proposal as was performed when generating the system recommendations.
- Provide explanations or justifications of various steps in the decision process.
- Capture human and automated decision making and the associated justifications for future review, training and forensic and trend analysis purposes.

2. TECHNICAL APPROACH

2.1 Natural Knowledge Representation for Decision Support

To enable decision support and decision making by *Sentinel*, we must capture and represent the knowledge that the human experts use in their decision making process. Expressing this knowledge in a way that is both easy for humans to understand and is machine executable is an essential prerequisite to automated decision making and explanation. The initial knowledge construction in *Sentinel* will be performed by knowledge engineers with the help of subject matter experts (SMEs) and will allow *Sentinel* to be useful out of the box. However, without continued maintenance, this knowledge can get out of date quickly as processes, technology and users change over time. Hence, *Sentinel* goes to great lengths to ease the task of knowledge representation and knowledge maintenance by the use of the following techniques:

1. **Knowledge representation using concepts that the users are familiar with.** *Sentinel* draws on multiple knowledge resources (ontologies) as the foundation for its vocabulary. For instance, *Sentinel* uses an ontology to represent concepts relating to aircrafts, another one to represent concepts relating to geopolitical entities, a third one to represent concepts relating to weapons carried on board aircrafts and so on. Yet other ontologies may capture the complexities of mission status and emergency exceptions. Note that many of these ontologies might already exist. Users can use third-party visual ontology building tools (such as Topbraid and Protégé) to develop these ontologies, and import them into *Sentinel*. By establishing *Sentinel*'s foundational vocabulary on these work-centered concepts we can remove an important barrier to knowledge representation.
2. **Representation languages that can naturally capture decisions that arise in practice.** While ontologies are an important ingredient of formalizing decision making policies, they are inadequate by themselves to capture the different kinds of decision making that arise in practice. For instance, a decision making policy such as "Consider pursuing the TST only if an asset is available that can travel at supersonic speeds and reach the TST within the deadline and if the asset is loaded with weapons capable of prosecuting the TST. In addition, the expected collateral damage assessment should be less than a certain threshold, and there should be no friendly troops in the area" refers to multiple ontologies — an ontology of aircrafts, an ontology of weapons and an ontology of military relationships. Such policies are not representable using ontologies alone. Other examples of decision making policies that go beyond the representational capabilities of ontologies include policies about established precedent or about exceptions.

   Sentinel formalizes such decision making policies by using a novel knowledge representation formalism which we call *decision rules*. These decision rules are similar to the rules used in rule based systems (see [1] and [2]), but have the following important distinguishing characteristics.

   a. Decision rules can include heterogeneous primitives from multiple ontologies and primitives that are defined by algorithms that run outside of the rule framework. For instance, to determine whether or not the primitive *NoFriendliesInRange* is TRUE when that information is not explicitly available might require obtaining the position of friendly forces within the region of interest, and running an algorithm to check that their planned path is sufficiently far from the region where the encounter with the TST is expected to take place.

   b. Decision rules can be time dependent, which makes it easy to state policies such as "This policy remains in effect for the duration of this operation". They can easily incorporate default assumptions.

Note that capturing decision making policies explicitly using decision rules allows them to be reviewed and modified, if necessary. Such rules can also be used for the training of new personnel.

3. **Rule Understanding and Validation.** Along with the introduction of new rules in *Sentinel* comes the need to ensure that these rules are correct, sufficient and consistent. *Sentinel* provides facilities to ensure the correctness and sufficiency of a set of newly entered rules by allowing the user to observe the system behavior on sets of data from previous missions that exercise these rules and verify that the rules are triggered appropriately. If not, the user can modify the rules and repeat the testing process until he is satisfied. Finally, *Sentinel* provides graphical tools for editing and visualizing rules and ontologies (similar to [3], [4],[5],[6]).
4. **Efficient maintenance of decision logic.** *Sentinel* maintains an effective separation between the representation of decision rules and the logic required to enforce such rules in order to provide the necessary system maintainability over time. In addition, to ensure that users are able to make changes to ontologies and rules as they evolve over time, *Sentinel* insulates the user from the need to know the underlying representation language. This is accomplished by the use of appropriate Graphical User Interface (GUI) templates that allow rules to be edited in an intuitive fashion.

2.2 Distributed Decision Making

Making the determination that a popup target is a TST and deciding to engage may be a collaborative effort between distributed entities, such as the Air Operations Center (AOC) and Airborne Warning And Control System (AWACS). Since the computing and manpower resources on board the AWACS are limited when compared to the AOC, pushing all the decision making into the AWACS is currently impractical. What is needed is some way of smoothly incorporating the decision making performed by the AOC and the decision making performed by the AWACS, or more generally, by several decision making nodes. Encoding decision making policies by decision rules easily lends itself to achieving this goal as follows. One first defines a set of "interface concepts" in the ontologies that serve to distribute the decision making. For instance, one such interface concept in an ontology aboard the AWACS can be "LegalOK", which means that there is no legal objection to pursuing the given TST. Checking whether or not LegalOK holds can be performed at the AOC (for instance, by using additional rules or by asking a human expert) and the result transmitted to the AWACS. Such a distributed decision making capability has several advantages:

1. Individual steps of the decision making can be performed at the nodes that are most qualified, and the results combined.
2. Delegating decision making to other nodes allows nodes that have less computational power and/or bandwidth to participate effectively in decision making.
3. The decision making can involve using processes outside the rule framework. For instance, determining the truth value of LegalOK might involve running a complex calculation to compute the expected path of a pursuing aircraft to make sure it does not intrude into enemy territory and querying a human expert about border treaties.
4. Default assumptions can easily be incorporated. For instance, if one can assume for a particular mission that one need not check to see whether there is any legal objection, this assumption can be incorporated by setting LegalOK to true by default for that mission.

2.3 Incremental Automation

For decision support systems to be used in practice, users must be able to check system recommendations and develop trust in them over a period of time. By explicitly representing concepts, *Sentinel* allows the user to drill down into the details of any particular decision making step. Such detailed visibility increases the human decision maker's confidence in the recommendations that *Sentinel* makes and also allows him to change an input on which a decision is made or override the results of a decision making step. *Sentinel* also provides capabilities that further support human decision making, such as making sure that all the relevant pieces of information have been taken into account and that all the relevant constraints have been satisfied.

In addition, *Sentinel* provides facilities for tracking system performance (i.e., how often recommendations are chosen versus overridden by users). This information can be used, through machine learning techniques or by explicit consent of the decision makers, in order to fine tune the ontologies and rules that define the decision making logic. Over time, such feedback enables *Sentinel* to improve the internal components that are involved in making decisions and will provide additional confidence that *Sentinel*'s decision making capabilities are adequate, thus further reducing the workload on the human decision maker.

3. ARCHITECTURE

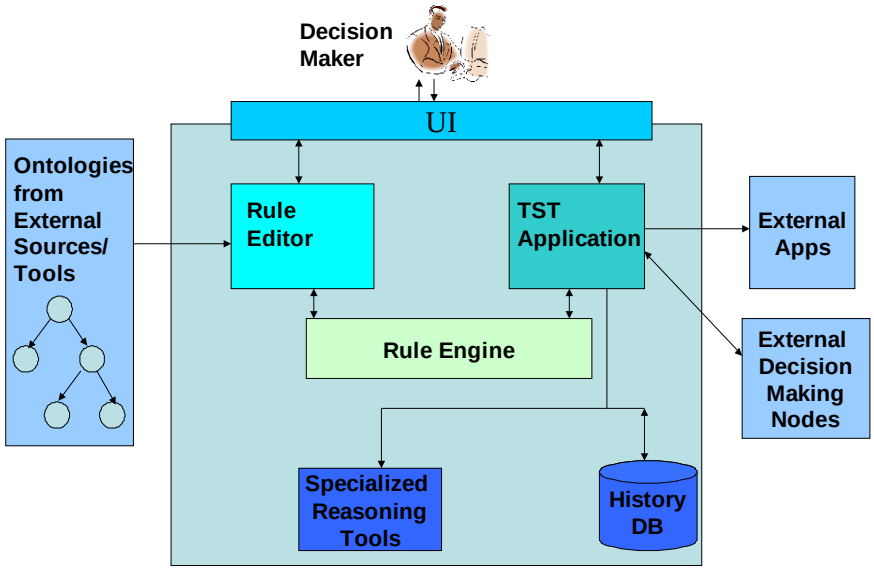

Figure 1: *Sentinel* Architecture

Figure 1 gives a simplified overview of *Sentinel*'s architecture. Ontologies can be created using visual ontology creation tools (such as TopBraid or Protégé) in W3C supported OWL standard (see [7] for details on OWL) and imported into *Sentinel*. These ontologies can serve as primitives in the decision rules used by *Sentinel* to represent decision making policies. The rule editor of *Sentinel* will insulate the *Sentinel* user from needing to know the details of the rule representation language while encoding decision policies as decision rules. The rule engine is the component that supports reasoning with decision rules. The TST application represents the particular decision support application (in this case, the one that helps users in the TST prosecution task) that is built using the decision rules. Note that the TST Application can call External Applications (such as an application for computing the weapon yield) and External Decision making nodes (such as an AOC node) to help its decision making process. It can also call specialized built-in reasoning tools (such as a constraint solver or a planning engine). The set of recommendations that are made by the TST application, as well as the set of choices made by the decision maker are stored in a History database for monitoring, training and rule improvement.

Let us now turn to a more detailed look at the TST Application component.

4. TST APPLICATION

The decision whether or not to pursue a TST is made using the rules of engagement (ROE) in effect. Once the decision to pursue the TST has been taken, the next question is: what asset should be used pursue it? For this decision, the relevant factors include (a) Whether the asset can reach the TST in time; (b) Whether the health of the asset (including the amount of fuel remaining) is adequate to pursue the target; (c) Whether the asset has weapons capable of prosecuting the TST; and (d) Whether the asset is involved in another higher priority mission. In addition, the TST Application must handle issues that arise in practice, such as (a) All the relevant knowledge might not be represented within the system (b)

An authorized human should be allowed to override the system recommendation (c) The system should be able to explain why it came up with a particular recommendation.

The TST Application uses rules, in combination with ontologies to represent policies that are used to decide whether a TST should be pursued and if so, what the appropriate assets are to pursue them. For instance, the policy rule "If an asset can reach the TST by the deadline, mark it as a possible candidate" is translated into a rule in the rule language that uses the concepts in the ontologies such as Assets, TSTs etc. as follows:

```
(defrule deadline-rule
        (Asset (id A) (type aircraft) (location L1) (speed S))
        (TST  (id T) (location L2) (deadline D))
        (test (withinDeadline A L1 T L2 D))
        =>
        (assert (candidate-capable-of-pursuit-in-time A T)))
```

This rule says that, given an Asset (call it A) with location L1 and speed S, and given a TST (call it T) at location L2 which needs to be engaged by the deadline D, if one can determine that A can reach T by the deadline, then assert that A is a candidate capable of pursuing T. Note that determining whether A can reach T by the deadline could be an involved calculation that takes into account the speeds and locations of A and T, the locations of threat areas, etc. Such calculations can be represented within the rule language itself or, if there is a preexisting function that performs such calculations (say via a web service), it can be called from within the rule itself. As pointed out before, one advantage of a rule based approach to decision making is that distributed decision making can be supported very easily. This is of special interest in the TST domain, where the decision whether to pursue a TST often requires cooperation between the AWACS and AOC. To express such dependencies in rules, we introduced special concepts called "interface concepts" into the ontology. During rule execution, a rule that includes such a concept will not fire until input is received from the AOC, but other rules that do not depend on the concept can proceed normally.

4.1 Interaction with the Application

Let us now examine how a user will interact with the TST application. After the user logs in, he is presented with a TST Mission board that shows the TSTs currently under consideration and their status, using color codes to draw the user's attention to those that are pending (Figure 2). The TSTs color coded with dark blue are the ones that are new and require immediate attention. The TSTs color coded with light blue have had assets committed to them, but the missions have not yet started. There are two other categories, those TSTs whose status is *ongoing* (meaning that assets have been assigned to them and the mission has started) and those whose status is *done*.

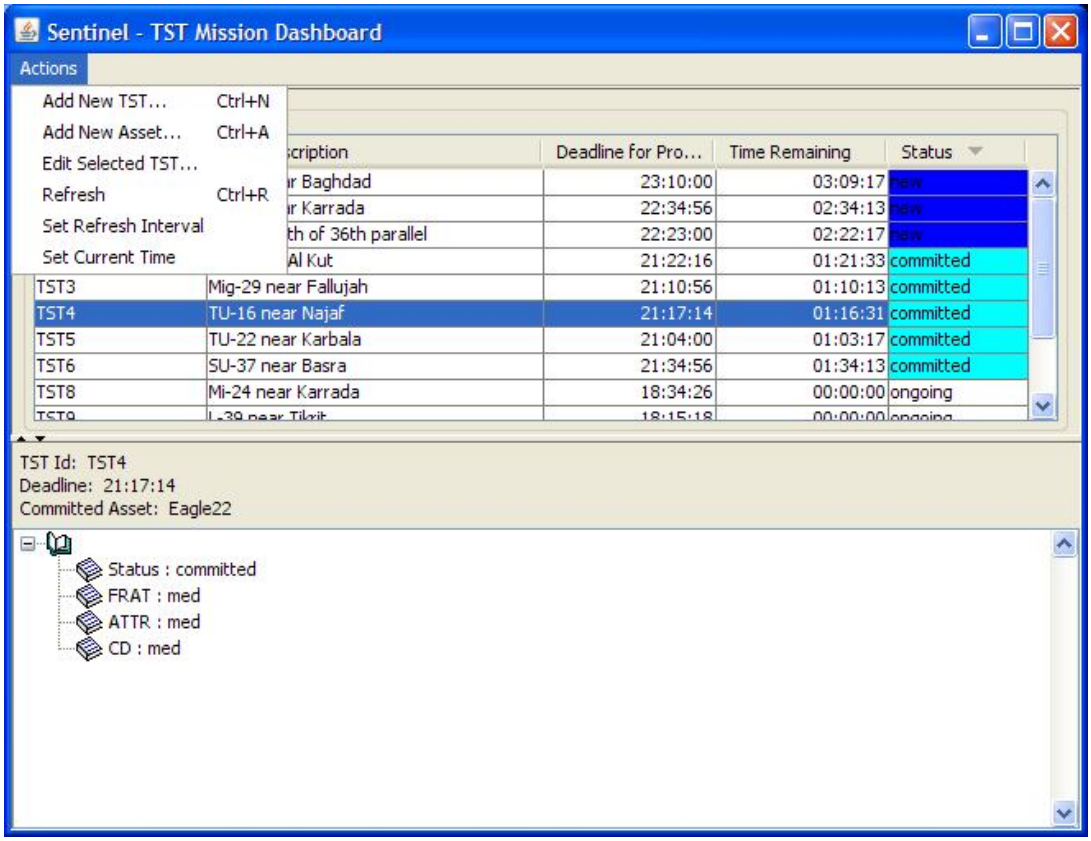

Figure 2: TST Mission Board

The user can perform several actions from this screen, such as adding a new TST, adding a new asset and editing a selected TST. He can also perform a refresh that might bring one or more TSTs or assets into consideration using data from an external system via a web services interface.

The user can double click on a TST to view the set of candidate assets capable of pursuing it, as shown in Figure 3. These candidates have been selected from the list of all possible assets by applying rules (such as whether the candidate asset can engage the TST by the deadline, whether all rules of engagement have been followed, whether the asset has weapon systems capable of engaging the TST and so on). Note that by default, we apply strict filtering, meaning that only assets that satisfy *all* the relevant rules are shown. These assets are then ranked using secondary criteria such as expected collateral damage. The computation of such criteria can be performed directly by the rule engine or by calling external applications for computing such criteria, when available.

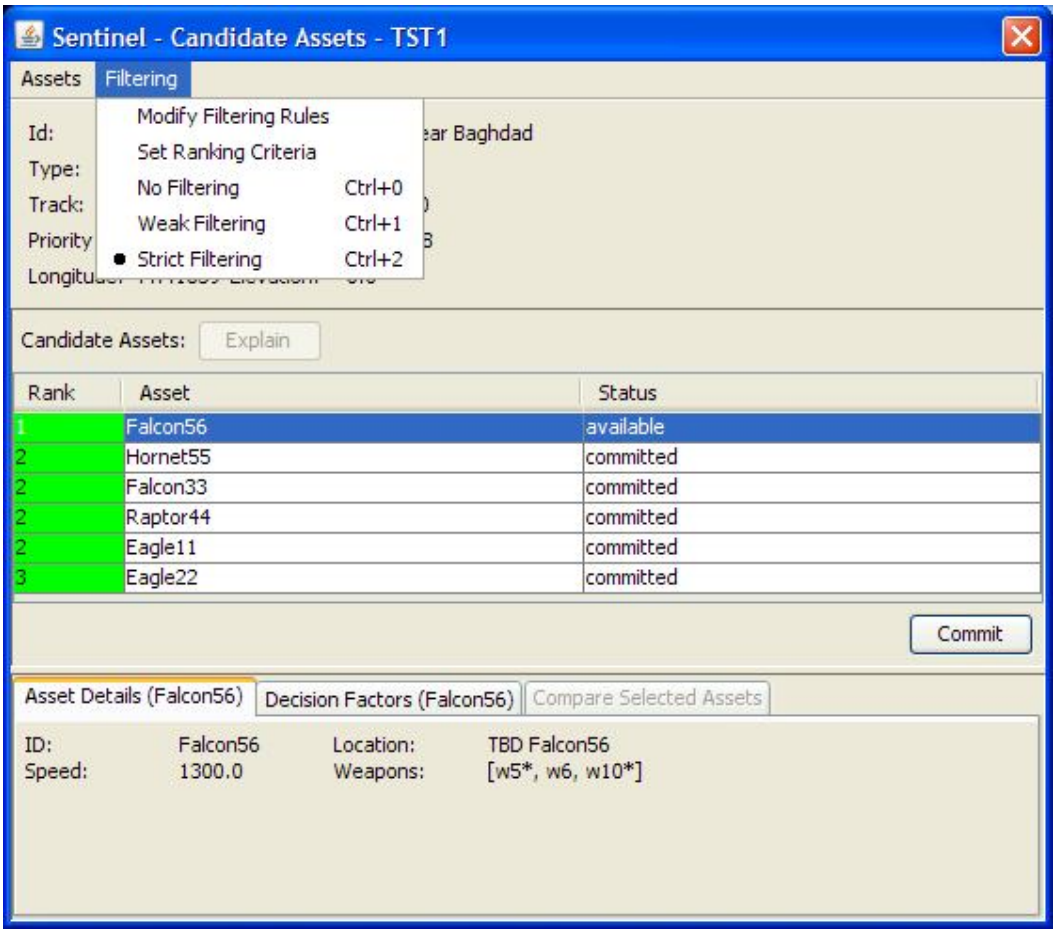

Figure 3: Candidate Assets for pursuing a TST

The user also has the ability to see exactly which rules have been violated or satisfied. Some users might want to use the secondary criteria differently to rank the assets. To accomplish this, the prototype provides a "Set Ranking Criteria" window (Figure 4) that allows the user to change the weights on the ranking criteria used to rank the assets.

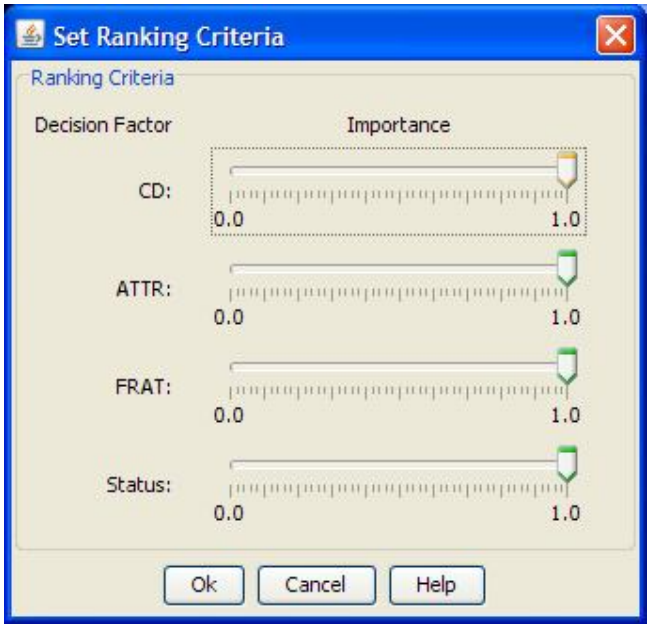

Figure 4: Set Ranking Criteria Window.

The user can also examine the details of the asset, or compare the assets on a number of dimensions (such as collateral damage, risk of fratricide, etc.). Once the user has made up his mind, he can commit to an asset by using the commit button.

Realizing that the modern battlefield is very dynamic and it is unreasonable to expect that the most updated information will always be available in the *Sentinel* system, our prototype provides the ability for the user to mark an asset as unavailable when information is received from the battle front to that effect. Once an asset is marked as unavailable, it will be excluded from all future planning unless the user sets its status back to available. Similarly, our prototype provides the ability to perform what-if reasoning with assets currently marked unavailable, anticipating that they are likely to become available soon (for instance, when scheduled repairs are completed).

Our prototype also permits the user to modify some of the underlying rules via an intuitive interface when the user has additional information that makes it clear that it would be beneficial to change the application of those rules. For instance, suppose the user comes to know that the TST under consideration is a supersonic aircraft and hence attention can profitably be restricted to assets above a certain speed threshold. This can be accomplished via the Modify Filtering Rules interface shown in Figure 5. This interface permits the user to modify a filtering rule in an intuitive fashion by, for instance, filling in some parameters. For example, if the user wants to focus attention on assets whose speed is greater than 1,200 mph, this is easily accomplished by setting the value in the textbox to 1,200 mph and clicking apply. Similarly, the user can easily modify other filtering rules to eliminate assets that do not have appropriate weapon types, assets that are too far away, etc.

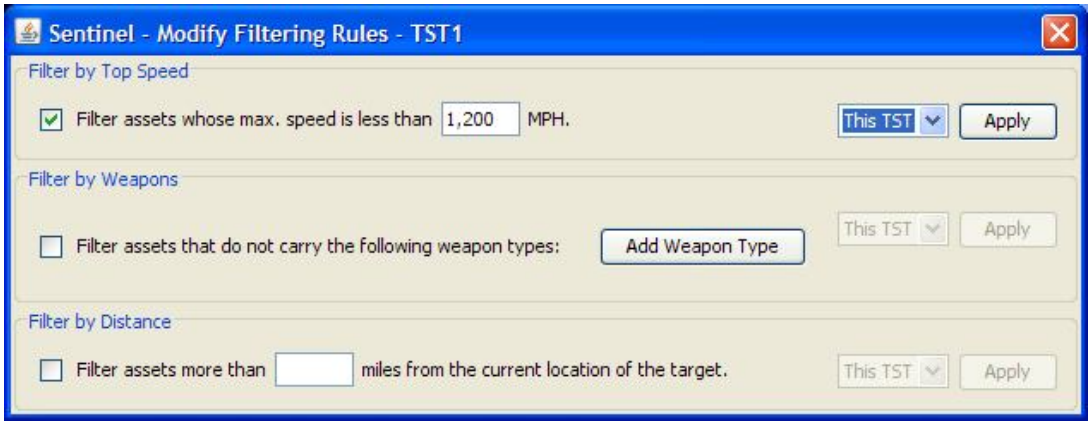

Figure 5: Modifying a Filtering Rule

5. CONCLUSIONS

In this paper we have described the technical approach, architecture and progress we have made in developing the *Sentinel* system – a system for supporting human decision making in the pursuit of time sensitive targets. *Sentinel* provides the means to easily represent and reason with the knowledge relevant in the decision-making process. Although we have presented this discussion in the context of a specific decision-making task for the sake of concreteness, it should be clear that *Sentinel*'s approach is general enough to support a wide variety of decision support tasks. We are currently in the process of deepening the inferencing capabilities of *Sentinel*, adding more explanation capabilities to it and improving the user-interfaces.

6. ACKNOWLEDGEMENTS

This project was funded through the Air Force Research Laboratory (AFRL) Small Business Innovative Research (SBIR) contract FA8750-08-C-0135. We would very much like to thank AFRL's Albert Frantz for his guidance and assistance in this project.

REFERENCES

[1] Girratano, J. and Riley, G., [Expert Systems: Principles and Programming], Fourth Edition, Course Technology; (2004)

[2] Hayes-Roth, F., Waterman, D., and Lenat, D. (Eds), [Building Expert Systems] Addison-Wesley, (1983)

[3] Mutton, P. and Golbeck, J., "Visualization of Semantic Metadata and Ontologies" Proceedings of Information Visualization, London, July 16-19,(2003)

[4] Pietriga, E., "IsaViz: A visual environment for browsing and authoring RDF models" Eleventh International World Wide Web Developer's Day (2002)

[5] Storey, M., Musen, M., Silva J., Best, C., Ernst, N., Fegerson, R., Noy N., "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge capture in Protégé". Workshop in Interactive Tools for Knowledge Capture (K-CAP-2001),(2001)

[6] Hopfner, M. and Siepel, D., "Visualizing Rules and Reasoning About Rules" Technical Report, Department of Computer Science, University of Wurzburg, July (2005).

[7] Allemang and Hendler, 2007 [Semantic Web for the Working Ontologist (Effective modeling in RDFS and OWL)] Morgan Kauffman, (2007)