Rules Based Decision Aid Framework

Executive Summary

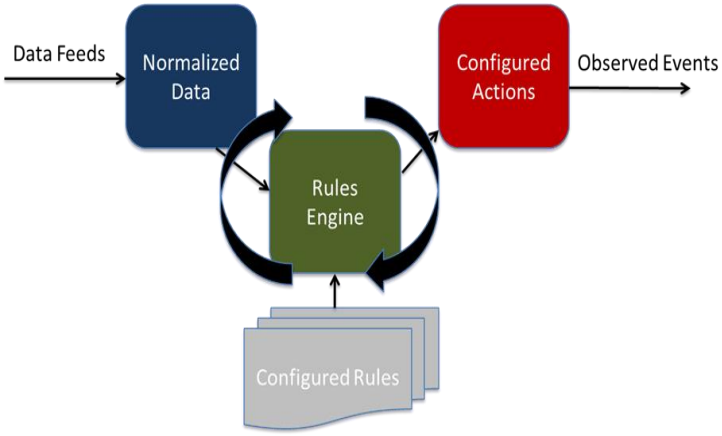
To further support automation in a Command and Control system, decision trees can be established to create an "If this, then that" model where disparate data relationships can be associated to drive rule executions, which in turn drive resultant actions. There exists many different Rules Engines in the commercial world, Drools being an example of one of them. I would like the team to research a Rules Engine, such as Drools, and prototype a framework instantiation of the product that would allow us to integrate it into a Command and Control system to facilitate decision aids execution. This framework should not be dependent on one specific set of data, rules or resultant actions, this way we could instantiate it multiple times for different needs.

Background

The fundamental goal of a Command and Control system, or any system, is to make timely/efficient decisions off of whatever set of data is available to support the operator in their day to day mission needs. It does this by collecting information, refining/associating/fusing that information, making decisions on that information and then cueing/visualizing the results to an operator. In many cases the information, by itself, is not valuable to the decision process, but when combined with other contextual data (time and space being the most prominent attributes), one could derive or infer results that could be extremely useful. These results could lead to better resource preservation, better resource coordination, better mission planning, faster object identification, faster kill chain response, better battlespace management, etc. All of these areas require a myriad of information and, in some cases, a tiered set of decision points to provide useful recommendations.

As one could expect, establishing a decision model for a command and control system is very contextual and there can be (and are) many different decision models in a system. Each of them performing their own unique set of functionality, in their own unique architecture on their own unique data feeds. This may be efficient, but when it comes to complex decisions, extensibility may be difficult. How easily would it be to add a new data feed to the existing decision making process? How hard would it be to change how the data is evaluated? Can I reuse the current logic and chain decisions together, if needed, to support branch decisions or higher level contextual decisions? The answer to all of these is yes, this can be done, but it may be expensive, it may break some existing logic, it may not be maintainable and it may not be something we can reuse in another part of the system. That's were a Rule-Based Architecture comes in.

A Rule-Based architecture can potentially alleviate some of these concerns and establish a more open/reusable framework for creating, scaling and tiering decision based functionality. A Rules-Based architecture is made up of three components: the Data, the Rules and the Actions. The data can be represented by a raw or normalized data feed, operator input or the output of another decision point. The Rules are represented by configurable artifacts that outline what scenario is being evaluated, with what data and what action to execute after evaluation. The actions are the configurable events to take place after a rule evaluation. After Rules are defined they are preserved for current or future use. Rules can be activated/deactivated at any time to adjust what scenarios are being evaluated. The Rules architecture can employ the Model-View-Controller Pattern to allow 1-n observers to register for events or data availability within a given framework or tier. This provides for an open model where observers can come and go through configuration without impacting the framework.



Task

Utilizing open source Rules Engines, evaluate and prototype a decision framework that can be utilized in a Command and Control system. The framework should not be directly coupled to a specific decision point or to the data in which it evaluates. Rather this framework can support the dynamic configuration of the following information:

- What information it will evaluate, assume this information is already available in the system
- Definition of the rule(s) to be evaluated
- What actions to take after successful rule execution
- The chaining of rules for branched or tiered decisions

We would like to show that this framework can be instantiated in different ways, using different configuration parameters, for different decision needs. In some cases, one rules framework instance may want to refer to another framework instance to execute higher order decision points.

Goals

- Evaluate more than one open source Rules Framework against the above set of tasking and determine which one provides the most flexibility while retaining performance
- Provide a whitepaper design document and prototype of the Rules Framework in action executing on multiple different decision points concurrently.
- Demonstrate branched decisions within a framework instance

- Demonstrate how two instances can be coupled/chained together
- Document the performance and scaling attributes of the Rules Engine against varying degrees of decision complexity and/or number of data sources
- Develop a simple UI to allow one to manage, configure and activate/deactivate a Rules framework instance.

The emphasis for this activity is on the frameworks flexibility and performance vs the visualization aspects.