**Project Design Document**

Project Title: Rules Based Decision Aid Framework

Distribution:

ASRC Federal Mission Solutions, Kimberly Davis

ASRC Federal Mission Solutions, Kevin Wainwright

ASRC Federal Mission Solutions, Christopher Barone

Rowan University, Professor Jack Myers

**Table of Contents**

Introduction…………………………………………………………….………....2

Purpose………………………………………………………………...………....2

System Overview………………………………………..………….…………....3

Roles and Responsibilities………………………………………………...…....3

Project References………………………………………………………..….….3

Technical Architecture……………………………………………..…………….4

Configuration Specification………………………………….…….…………….5

Solution Design Specification……………………………….….…...……...…..6

Terms and Definitions…………………………………..……………………..…9

Revision History…………………………………………………..…..…………..9

**Introduction**

This System Design Document has been created to outline the proposed system design for new ASRC Federal Mission Solutions Rules Based Decision Aid Framework (RBDAF). The RBDAF is intended to replace a Command and Control system but to support automation in a one. By designing, testing, and deploying the RBDAF, ASRC Federal Mission Solutions will be able to integrate this solution into Command and Control systems to facilitate decision aids execution. This document and the technical specifications listed herein comply with like nobody’s technical standards and infrastructure.

**Purpose**

This section should provide a high-level description of the purpose of the System Design Document. This may include a description of how the System Design Document relates to organizational goals and/or objectives and how the new system will meet those goals and objectives.  
 The purpose of this System Design Document is to provide a description for how the new MMS will be constructed. The Systems Design Document was created to ensure that the MMS design meets the requirements specified in the MMS project requirements documentation as well as the Acme Corporation’s Executive Bulletin referencing improvements to existing maintenance management practices and tools. The System Design Document provides a description of the system architecture, software, hardware, database design, and security.

The purpose of this System Design Document is to outline the design of the RBDAF and how it will be constructed. The System Design Document was created to ensure that the RBDAF design meets the requirements specified in the project requirements provided by ASRC Mission Federal Solutions.

The main purpose of the document is to:

* Provide detailed functionality which will be provided by each component or group of components in the Rule Based Decision Engine.
* Show the interaction between the components in the system.

**System Overview**

The RBDAF architecture is made up of three components: data, rules, and actions. A rules engine (such as Drools -- the rules engine used in our system) will process the data as well as already configured rules and provide actions, not unlike a simple if-else statement.

The RBDAF system will be utilized in command and control systems as well as in many different applications with various types of data. By that accord, the model-view-controller pattern is best suited for this application as it will provide a GUI supplemented for any application and at the core of it all it simple just makes sense to use it to segregate the code of the data, rules, and actions through this pattern.

**Roles and Responsibilities**

The IT Solution Delivery Lifecycle Procedure, describes the Roles and Responsibilities for developing verifying, and implementing a Solution. There are no additional roles specific to this document

**Project References**

The RBDAF is being designed by the Rowan University software engineering Wolves team for Mission Solutions Engineering, who in turn are providing requirements, direction and bi-weekly product reviews. The following list of references show what is required for the functionality of the RBDAF, and what Mission Solutions Engineering requires from team Wolf.

* ASRC Federal Mission Solutions: User Story Done Criteria ver. 0.6, September 2016
* ASRC Federal Mission Solutions: Rules Based Decision Aid Framework Description, September 2016
* JBoss: Drools Documentation version 6.0.1 Final: https://docs.jboss.org/drools/release/6.0.1.Final/drools-docs/html\_single/

**Technical Architecture**

**Hardware:**

The RBDAF is based on existing hardware that the Rowan University software engineering teams have access to. Although each student has varying hardware, the software is expected to run on computers with specifications similar to the list below:

//Wasn’t sure what exactly to write here, since we don’t have access to MSE’s //hardware, I just wrote my computer’s specs as an example. Feel free to change of //course.

* OS: Microsoft Windows 10 professional
* System Type: x64-based PC
* Processor: Intel Core i7-5500U CPU @ 2.40GHz, 2401 Mhz
* RAM: 8.00 GB
* 1x 1TB hard drive

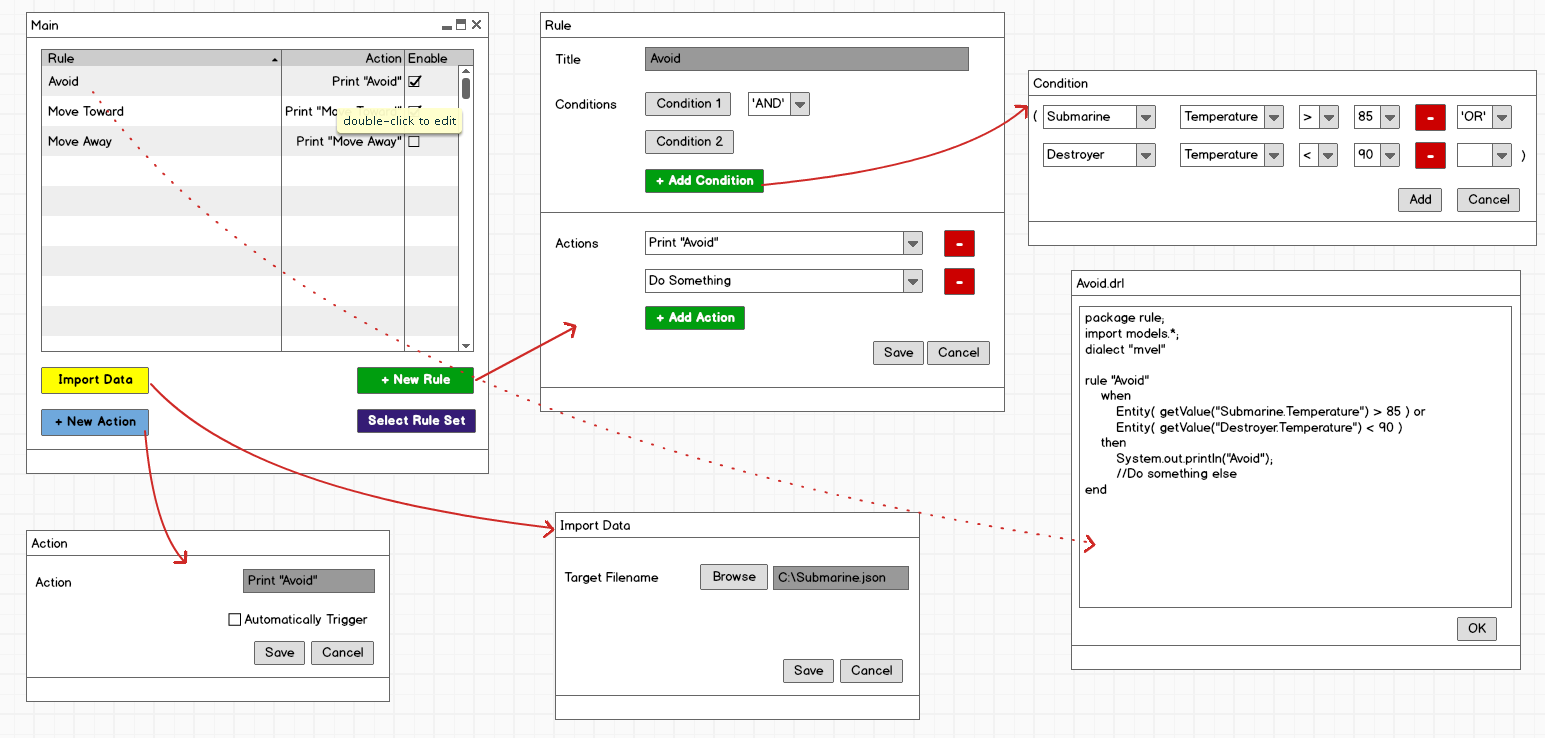
**Software:**

The RBDAF is being designed in Java with assistance from the Drools rule engine plugin. The software is able to take in data from various external file types, run that data with numerous user created rules and generate an outcome based on the association between the two. The main components that make up this framework include data input, rule object/Drools drl creation, and data association with rules. The list below describes these components in more detail.

* Import data module: This component includes a user interface which will allow the user to import data from files located in a local file system and create data objects from it. The module includes the ability to detect various file types including txt, JSON and csv and is open to extensibility by Mission Solutions Engineering.
* Rule Object/Drools drl creation module: This component provides the user with a UI that allows them to create new rule objects, and put them together to make the contents of a fully functional drl file with as many arguments as necessary. It both provides the user with enough simplicity anyone to make rules as well as a text editor for drools experts to make drl files themselves.
* Data association module: This component takes previously created data objects and inserts them into the selected user constructed rules in order to create actions based on the situation. These actions can perform various tasks which can range from simply alerting a user, to more complex situations such as firing off other rules (rule chaining).

**Configuration Specification**

When applicable, the configuration specification should be a technical design detail that records configuration values. Any operational description should be limited to clarification or refinement of info presented within the design specification. Diagrams of components may be used to illustrate relationships. Include any configuration information required to properly setup the Solution for use.   
This may include but is not limited to the following:  
Required operating system configuration, for example, regional settings and date format.  
Required software module setup parameters  
Configuration parameters for any associated utility programs, for example, report tools, data mapping/transfer tools.



**Solution Design Specification**

The design of the solution works in such a way that, for the user, the front-end application presented would closely resemble the structure of the Drools Rules Engine. This allows the user to envision a clear mental-model of what is occurring in back-end of the solution as well as the Drools framework itself. The GUI, as seen in a wire-frame model prior to this section, prompts the user the select a rule set, import data, create a new action, and create a new rule. A Drools rule has only three components that the user will have to set in the solution: a title, conditions to check, and actions to fire if these conditions are logically true.

Data being utilized by the solution can be an ambiguous type or set of types that will be handled by a JSON file creator and CSV file reader. The RBDEF solution is meant to be able to read in any type of data from things such as sensors, databases, text files, social-media, etc. Therefore, the solutions ability to perform condition checking on any data is resolved by JSON which converts it to a format called Entity in the solution.

The Entity is a representation of the data, a value pair with a title and a HashMap of data, allocating the name of a data value to a data map. For example, there could be an Entity for a vehicle. This Entity has the name “vehicle” and its HashMap of data could be states of any of the vehicle’s attributes, such as “Tires - Inflated” or “Gasoline Level - Empty” where “Tires” and “Gasoline Level” are attributes and “Inflated” and “Empty” are the states.

**Module Description**

*Controllers*

ActionController - The purpose of this class is to create an action object using the

parameters passed in

RuleActivation - The purpose of this class is to either receive the drl file(s) the user wants to run data through or create the file from scratch using Rule object data and strings.

EntityController - The purpose of this class is to create and add entities to the collection the system reads from, as well as retrieving entities and their keys/values

RuleController – The purpose of this class is to add the various components of a Drools rule, making a file containing the Drools rule, adding a rule to a collection, and firing rules (firing all of them)

*Exceptions*

ActionException - The purpose of this class is to Create an Exception Caused from Action Class

*Helpers*

CreateDroolsFile - The purpose of this class is to create an empty Drool file to store Rules.

CSVReader - The purpose of this class is to read a csv file which stores the data.

*Models*

Action - The purpose of this class is to create an action

ActionList - The purpose of this class is to create a list which stores all the Actions.

ConditionalElement - The purpose of this class is to create a Condition.

ConditionalElementList - The purpose of this class is to create a list which stores all the Conditions.

Constraint - The purpose of this class is to create a constraint for condition.

ConstraintList - The purpose of this class is to create a list of constraints.

Entity - The purpose of this class is to create an Object which stores all the data associated with it, including the data, type of the data and value of the data.

Location -

LogicalConjunction - The purpose of this class is to describe the relationship between constraints.

ObjectType - The purpose of this class is to describe the Type of the data(Int, String,..).

Operator - The purpose of this class is to describe the operations needed to perform between constraints and conditions.

Rule - The purpose of this class is to create a Rule which has name, condition and action for a rule.

*Rules*

The purpose of this module is to store all the rules files needed for the program to run.

*Services*

ActionCollectionService - The purpose of this class is to create a collection of Actions.

Collectable – The purpose of this interface is to serve as the skeleton for Entities that will be put into collections

EntityCollectionService - The purpose of this class is to create a collection of Entities.

FileWriterService - The purpose of this class is to write the actual drools files and save them to a path relative to the application.

FileReaderService - The purpose of this class is to read data files which are in JSON and CSV format and convert it to Entity.

InputParserService - The Purpose of this class is to parse file.

ObjectCollectionService - The purpose of this class is to create a collection of data object

RuleCollectionService - The purpose of this class is to create a collection of Rules.

RulesReader - The Purpose of this class is to read a Drool file.

SerializationService - The purpose of this class is to give the user the ability to pass and serialize/deserialize an object.

*Views*

The java classes contained in the views module are the different components used in the GUI along with accompanying forms for each class component.

**Terms and Definitions**

Below is a comprehensive table listing all acronyms listed throughout this document.

|  |  |
| --- | --- |
| ***Acronym*** | ***Definition*** |
| ***RBDE*** | *Rules Based Decision Engine* |
| ***RBDAF*** | *Rules Based Decision Aid Framework* |
| ***GUI*** | *Graphical User Interface* |
| ***COTS*** | *Commercial off-the-shelf* |
| ***PBI*** | *Product Backlog Item* |
| ***MSE*** | *Mission Solutions Engineering* |
| ***JSON*** | *JavaScript Object Notation* |

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Version Date** | **Revision** |
| 0.1 | November 18, 2016 | Initial Release |
| 1.0 | December 16, 2016 | Rule chaining addition |