# SRI DESIGN DOCUMENT

**OUTLINE OF PREDICATE:**
Base class of which both Fact and Rule inherit from.

| **Name** | **Description** |
|---|---|
| Components | vector of strings that represent relevant actors |
| Name | String name identifier |
| evaluate(vector<string> c) | returns boolean value of validity of Predicate |
| toString() | returns string representation |
| Predicate(string s, vector<string> a) | Constructor |

**OUTLINE OF FACT:**
Basic building block of inferencing. Has a name that identifies the relationship between two actors, of which are stored in a string vector. When evaluated, returns true if found in the KB and false otherwise.

| **Name** | **Description** |
|---|---|
| Components | vector of strings that represent relevant actors |
| Name | String name identifier |
| evaluate(vector<string> c) | returns boolean value of validity of Fact |
| toString() | returns string representation |
| Predicate(string s, vector<string> a) | Constructor |

**OUTLINE OF RULE:**
Rules are comprised up of Predicates, of which can be either Facts or other Rules. A Rule is valid if its required components are valid.  If the rule is valid, add the relevant fact to the KB and return true.

| **Name** | **Description** |
|---|---|
| Components | vector of Predicates that must evaluate to true if rule is to be valid |
| Name | String name identifier |
| evaluate(vector<string> c) | Checks validity of rule by checking validity of all of its components (can be either Facts or other Rules) |
| enact(vector<string> c) | invokes evaluate() and adds relevant Fact to KB if rule is valid. |
| toString() | returns string representation |
| Rule(string s, vector<int> ops, vector<string> a) | Constructor (ops is a vector of ints that act as a switch for AND and OR operators to use) |

**OUTLINE OF RULE BASE:**
Holds references to each Rule, along with methods to manage them.  Is a part of "Working Memory."

| **Name** | **Description** |
|---|---|
| rules | Vector of rule pointers.  Contains all rules in working memory. |
| Add(Rule * r) | Adds rule to the rule base |
| Remove(Rule * r) | Remove rule from the rule base |
| RB() | initializes empty rule base |
| toString() | returns string representation of all rules into a string so it can be printed |
| Load() | take input from an SRI file and load all the rules into the rule base |
| Dump() | Saves all the facts into an SRI file so it can be used for input later |

**OUTLINE OF KNOWLEDGE BASE:**
Holds references to each Fact, along with methods to manage them. Is a part of "Working Memory."

| **Name** | **Description** |
|---|---|
| FactsDirectory | std::map<std::string, Fact*> that holds fact pointers. Keys are strings representative of similar relationships |
| Add(Fact * f) | Adds fact to the knowledge base |
| Remove(Fact * f) | remove fact from the knowledge base |
| KB() | initializes empty knowledge base |
| toString() | returns string representation of all facts into a string so it can be printed |
| Load() | take input from an SRI file and load all the rules into the rule base |
| Dump() | Saves all the rules and facts into an SRI file so it can be used for input later |
| Fetch(std::string n, vector<string> actors) | looks for fact by name and actors and returns a pointer to it |
| Find(std::string key) | returns the vector of fact pointers associated with the key |

**OUTLINE OF QUERY**

The “Thinking” function that drives inferences. “Binds” actors together into combinations, which are then passed into each rule in the RB. The rules recursively evaluate their validity and populate the KB as necessary. Functions are put in place to not recur into duplicate combinations.

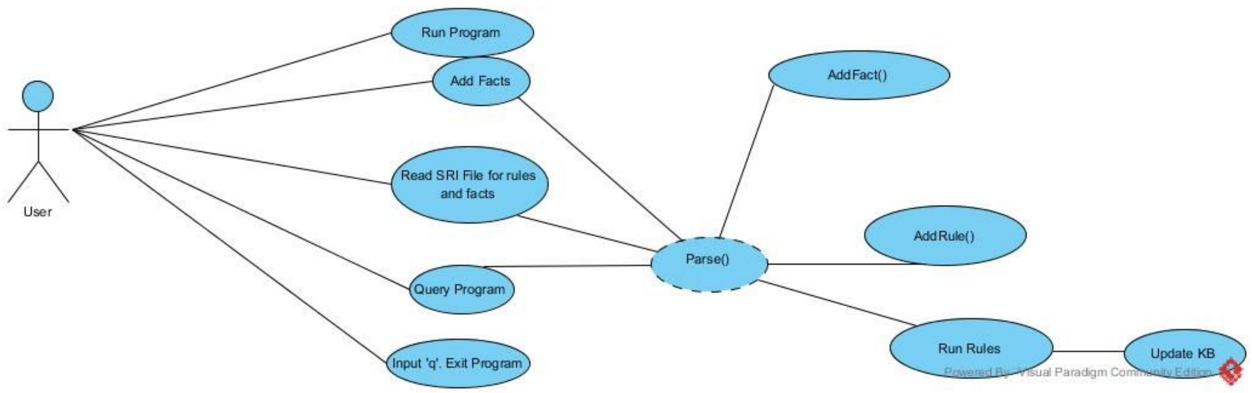
| **Name** | **Description** |
|---|---|
| kb* | Pointer to a KB object |
| rb* | Pointer to a RB object |
| listFact(KB * knowledge, string factKey) | Returns a pointer to a list (vector) of Fact pointers with a similar relationship from the Knowledge Base. |
| concatenate(vector <Fact*>* a, vector<Fact*>* b) | Helper function that will get rid of duplicates. Uses function preventDupes() |
| printResults() | Prints results of an inference run |
| Inference() | “Binds” actors together into combinations, which are then passed into each rule in the RB. The rules recursively evaluate their validity and populate the KB as necessary. |

**ASSUMPTIONS:**

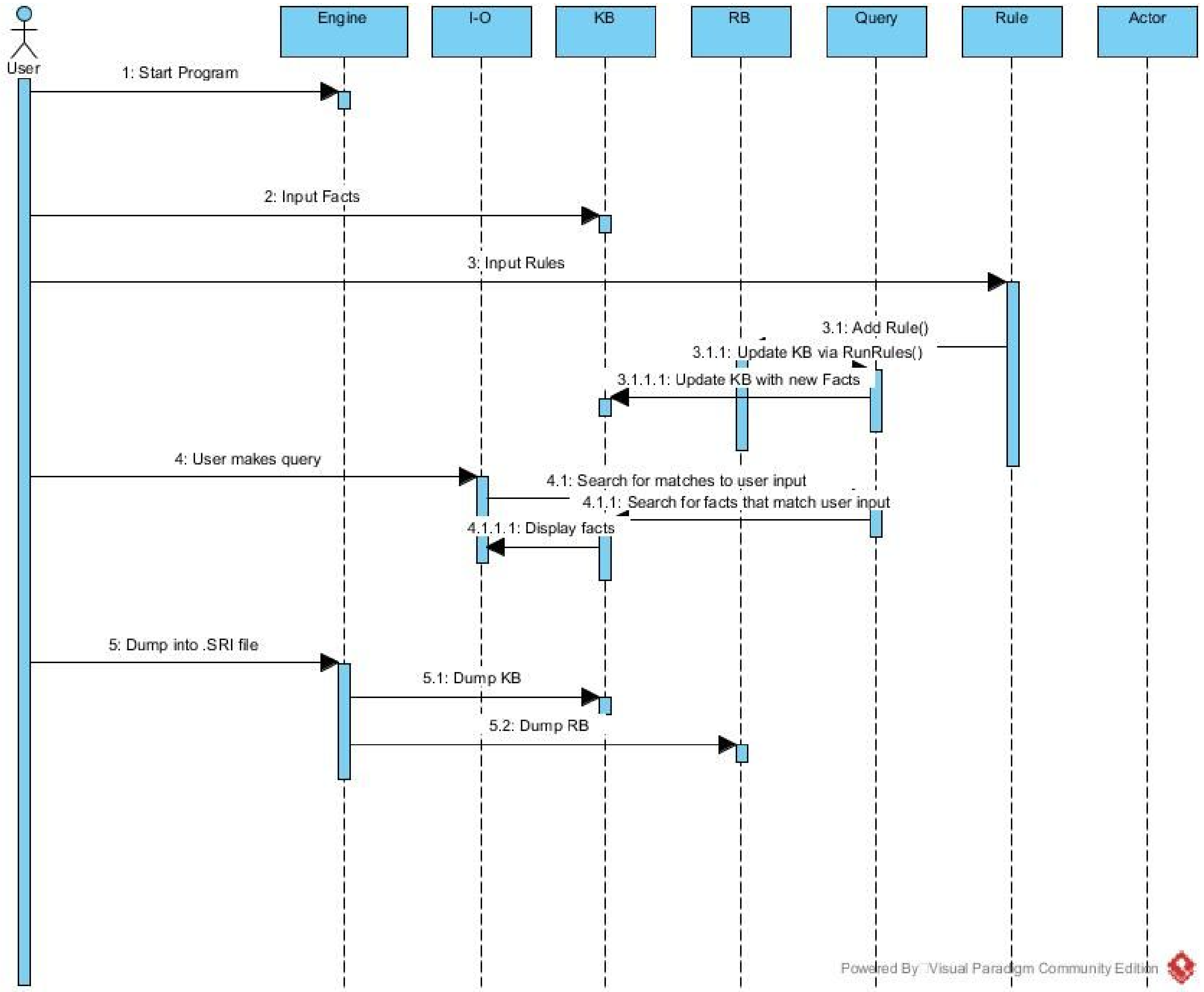
We are assuming the user is feeding the program proper input.
Rules are comprised of existing Facts and/or Rules already in working memory
A Fact is true if it can be found in the Knowledge Base

To compile:
Use: “make” without quotes in the root directory.

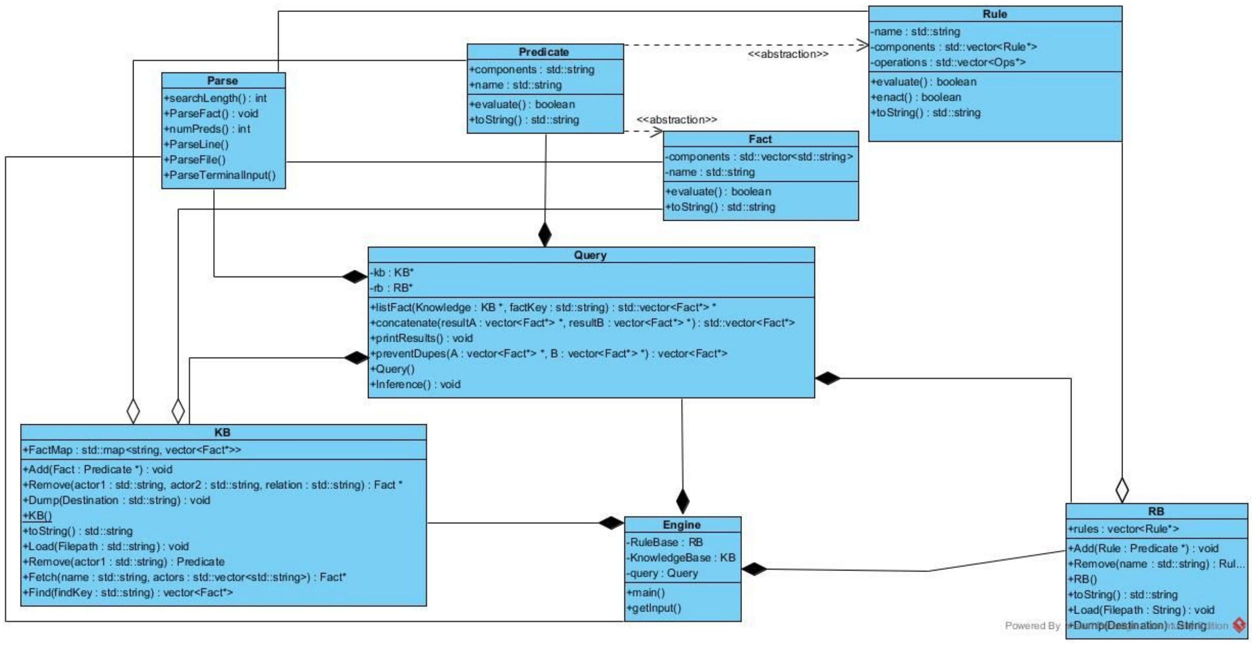
## USE CASE DIAGRAM:



## SEQUENCE DIAGRAM


User
Engine
I-O
KB
RB
Query
Rule
Actor
1: Start Program
2: Input Facts
3: Input Rules
3.1: Add Rule()
3.1.1: Update KB via RunRules()
3.1.1.1: Update KB with new Facts
4: User makes query
4.1: Search for matches to user input
4.1.1: Search for facts that match user input
4.1.1.1: Display facts
5: Dump into .SRI file
5.1: Dump KB
5.2: Dump RB
Powered By Visual Paradigm Community Edition

## CLASS DIAGRAM:


Rule
-name : std::string
-components : std::vector<Rule*>
-operations : std::vector<Ops*>
+evaluate() : boolean
+enact() : boolean
+toString() : std::string
Predicate
+components : std::string
+name : std::string
+evaluate() : boolean
+toString() : std::string
<<abstraction>>
<<abstraction>>
Parse
+searchLength() : int
+ParseFact() : void
+numPreds() : int
+ParseLine()
+ParseFile()
+ParseTerminalInput()
Fact
-components : std::vector<std::string>
-name : std::string
+evaluate() : boolean
+toString() : std::string
Query
-kb : KB*
-rb : RB*
+listFact(Knowledge : KB *, factKey : std::string) : std::vector<Fact*> *
+concatenate(resultA : vector<Fact*> *, resultB : vector<Fact*> *) : std::vector<Fact*>
+printResults() : void
+preventDupes(A : vector<Fact*> *, B : vector<Fact*> *) : vector<Fact*>
+Query()
+Inference() : void
KB
+FactMap : std::map<string, vector<Fact*>>
+Add(Fact : Predicate *) : void
+Remove(actor1 : std::string, actor2 : std::string, relation : std::string) : Fact *
+Dump(Destination : std::string) : void
+KB()
+toString() : std::string
+Load(Filepath : std::string) : void
+Remove(actor1 : std::string) : Predicate
+Fetch(name : std::string, actors : std::vector<std::string>) : Fact*
+Find(findKey : std::string) : vector<Fact*>
Engine
-RuleBase : RB
-KnowledgeBase : KB
-query : Query
+main()
+getInput()
RB
+rules : vector<Rule*>
+Add(Rule : Predicate *) : void
+Remove(name : std::string) : Rul...
+RB()
+toString() : std::string
+Load(Filepath : String) : void
+Dump(Destination) : String
Powered By