Katerina Chinnappan, Jakrarin Simrakut, Traik Zeid, Nathan Monahelis

CMPS 109 - Final Project Phase 3 report

What works:
Loading and printing rule and fact. Dropping Fact & rule, dump should work(hopefully). Inferencing a fact works as well. Inferencing rule not yet. Not finished with threading. Will get everything done by phase 4 ☹ Thanks

Class SRI:
SRI is our main class. It implements the commands, load, dump, drop(rules and facts), and inference. Inference is yet to be completed. The SRI class uses the KnowledgeBase and RuleBase pointer objects.

```
Void SRI::inputLine(stringStream &)

Void SRI::addFact(string)

Void SRI::addRule(string)

Void SRI::load(string) // It will then parse the SRI file line by line
//adding rules or facts into their respective databases.

Void SRI::dump(KnowledgeBase *kb,RuleBase *rb) // It will access the
current facts and rules defined //in the runtime KB and RB and then
save them to the SRI file given.

Void SRI::dumpRF(ostream &os,KnowledgeBase *kb, RuleBase *rb)

Void SRI::drop() // This will invoke a KB/RB method depending on what
//is being dropped. class KB : public Operator

Vector<map<string,string>> inferenceFact(string, vector<string> &);

Vector<map<string,string>> inferenceRule(string, vector<string> &);

map<string,vector<string>> findRule(string, int);

Void SRI::Inference(string query) //This will print the results of the
//given query to the terminal. Inference
//will have an option to declare the results of the query under a fact
//with a given variable name.
```

Class KnowledgeBase:

The KnowledgeBase class contains public string map of Fact* vectors which will hold the facts. Public methods such as the findFactAssociation which returns true if the association of a certain fact is found or vice versa.

```
bool KnowledgeBase::findFactAssociation(Fact * fact)

void KnowledgeBase::AddFact(Fact * fact) //add a fact to the
//KnowledgeBase Dictionary

void KnowledgeBase::dropFact(string param)//drop the fact from the
//FactDictionary

KnowledgeBase::~KnowledgeBase() //destructor
```

The KnowlegeBase class uses a Fact pointer object.

Class RuleBase:
The RuleBase class contains a public string map of Rule* vectors. Just like in KnowledgeBase, we can add and drop a rule we specify. The RuleBase class contains a Rule pointer object.

```
RuleBase::RuleBase()

void RuleBase::AddRule(Rule * rule)

void RuleBase::dropRule(string param)
```

Class Parse:
The Parse class will open up a specified .sri file and parse it. Basically, take it apart and separate the Facts from the Rules as well as process the logical operators.

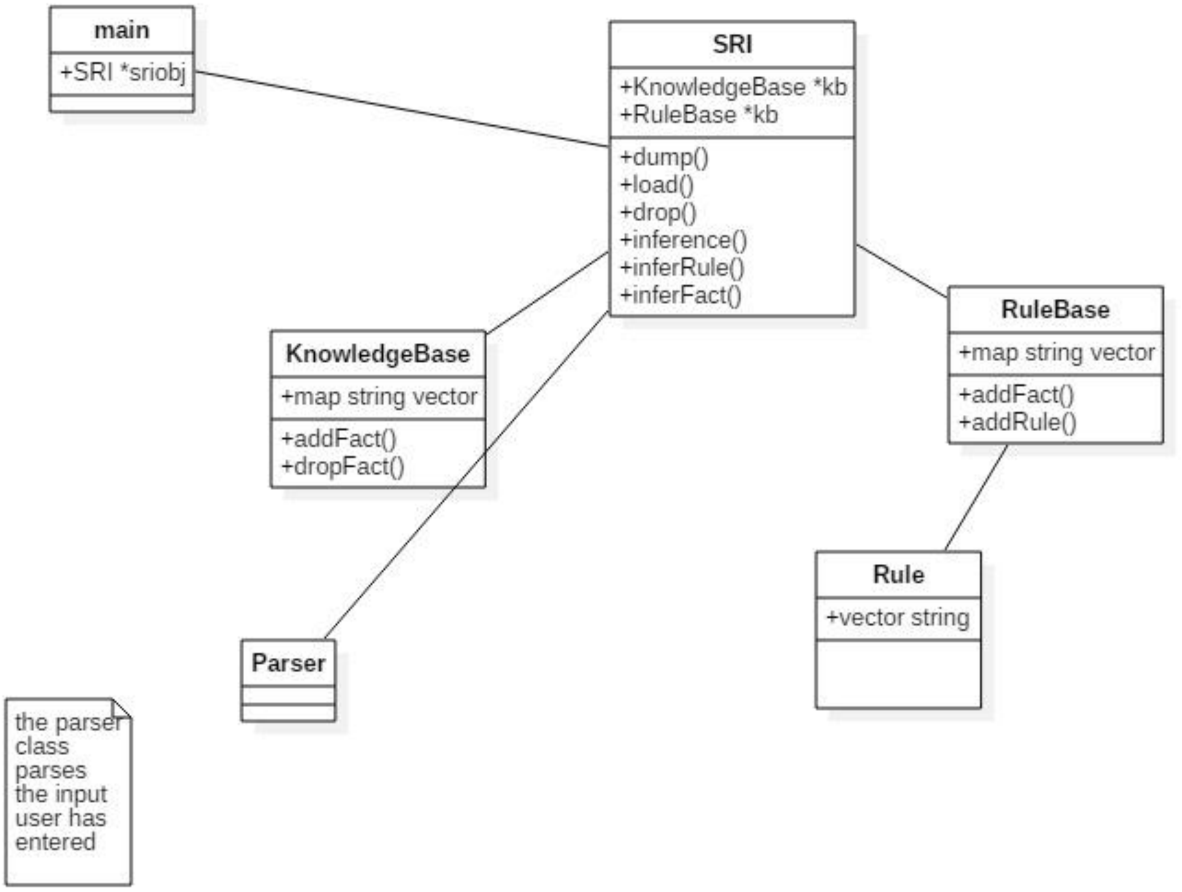
```
Bool getType(line) // determine fact/rule true=fact false=rule

String getFactAssoc(line) // returns fact Assoc

String  getRuleAssoc(line)//returns rule Assoc

Vector getFactParam(line)//returns a vector of string parameters

Vector getRuleParam(line)// returns a vector of string parameters

String getGateLine(line) // returns AND or OR of line
```

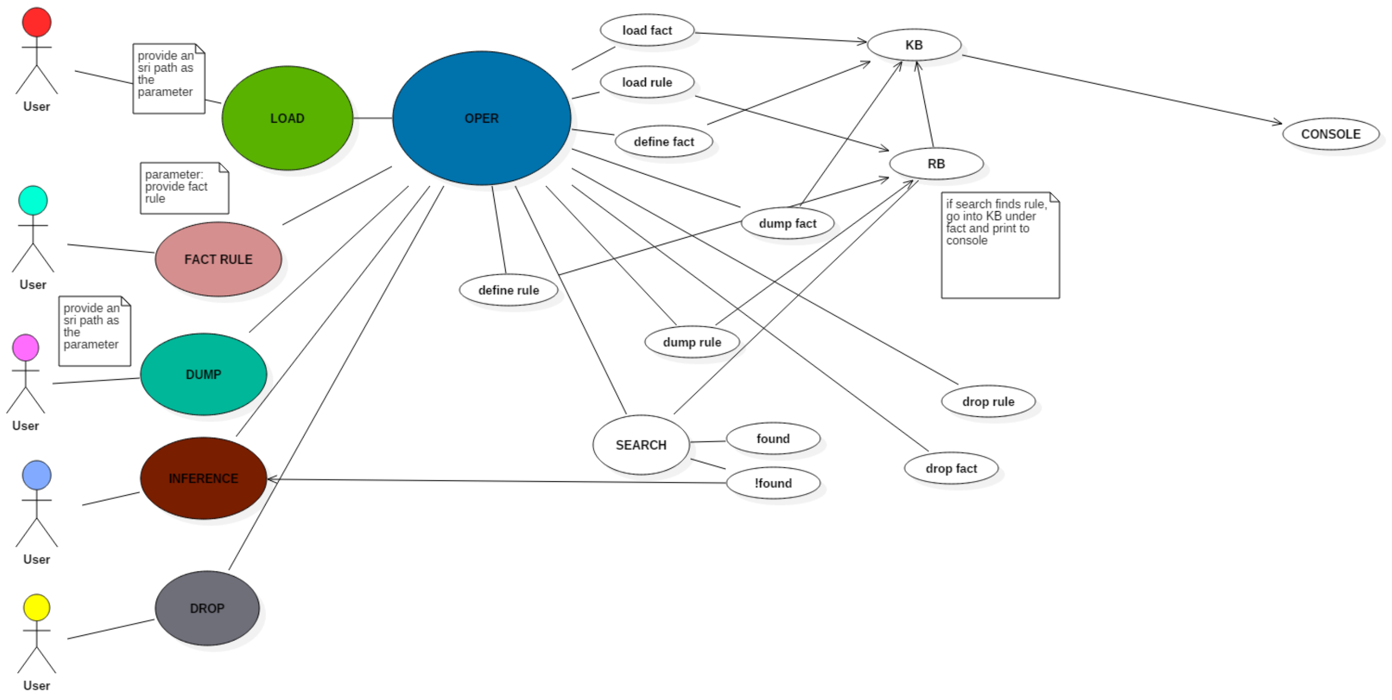
Main.cpp
This is the main function which has one SRI pointer object. Provides the user with a menu, takes in user input and performs the operations.

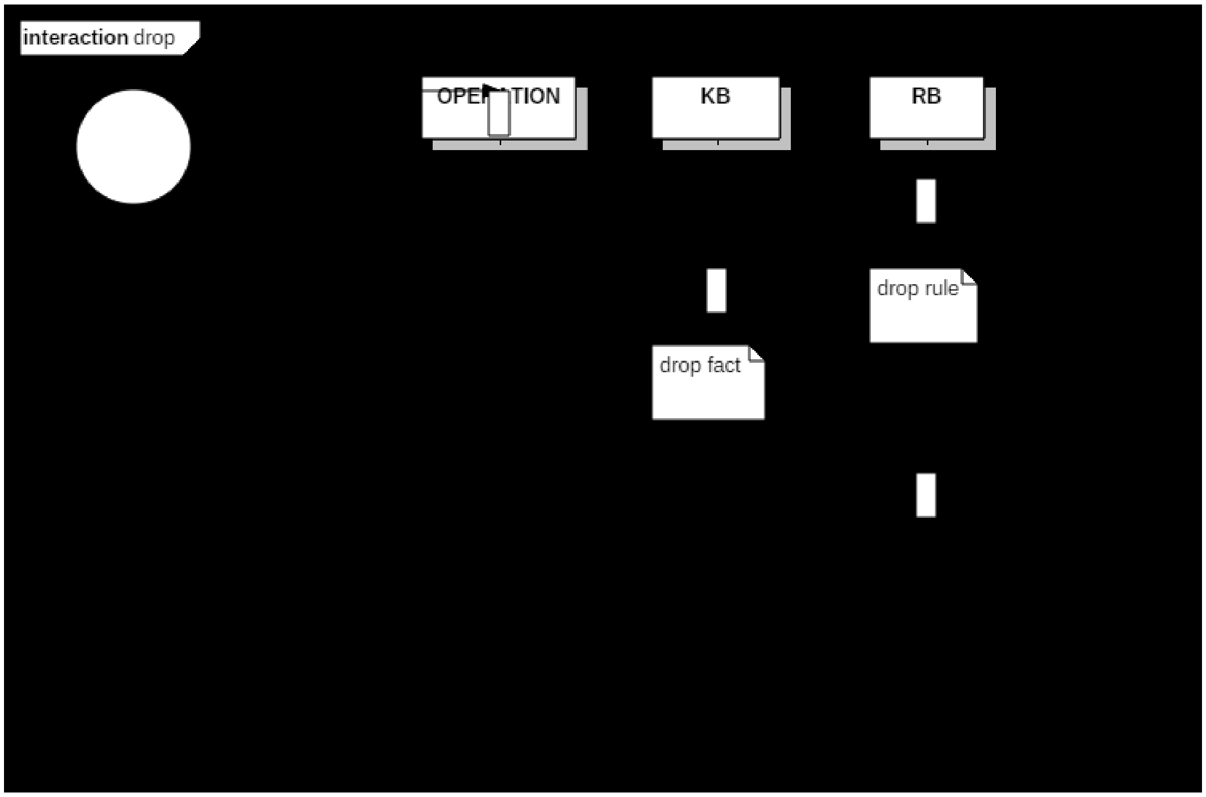
Common_headers.h
This is a header file which includes ALL C++11 headers, and std::functions to make life easier.

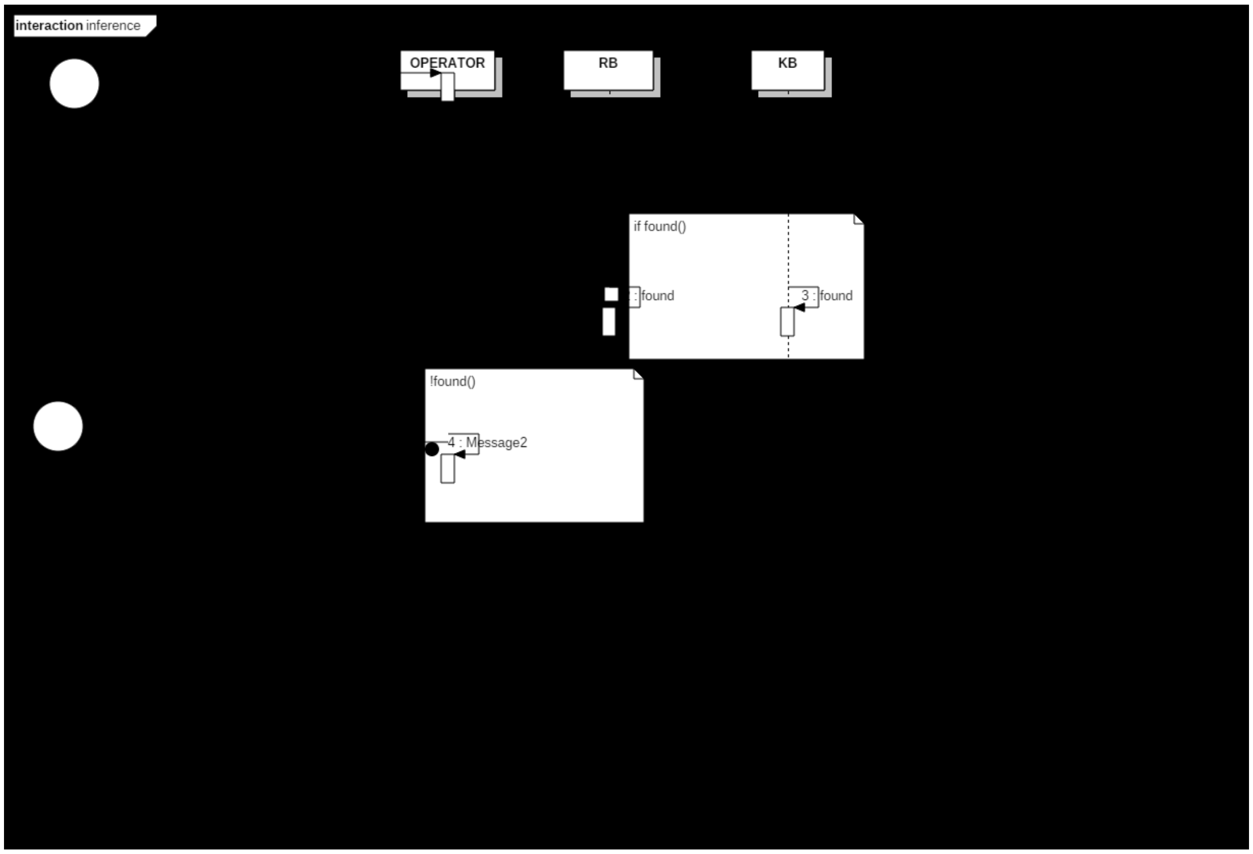
Makefile
Getting errors for “auto” keyword, not sure why. Please try to compile with g++

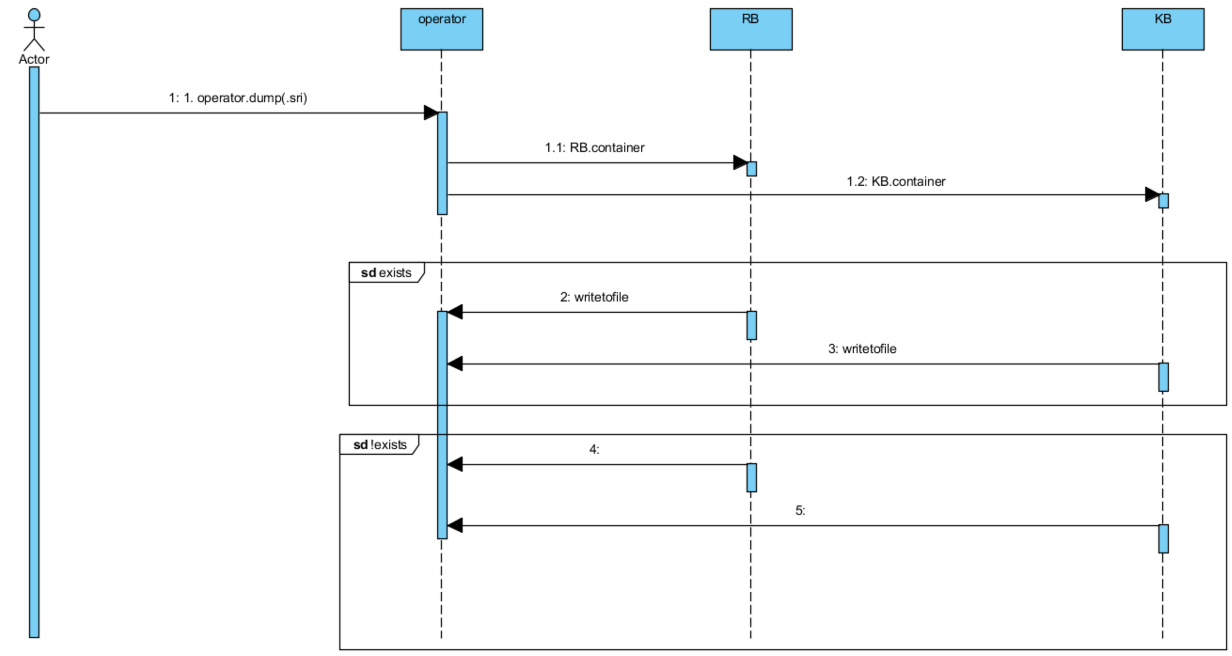
Class Diagram

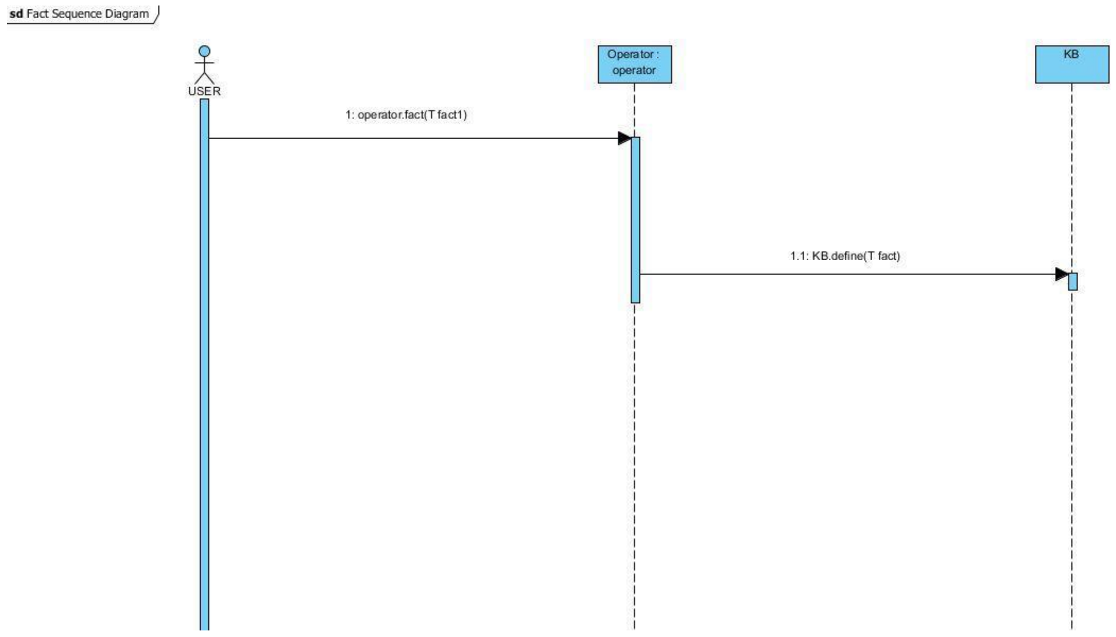


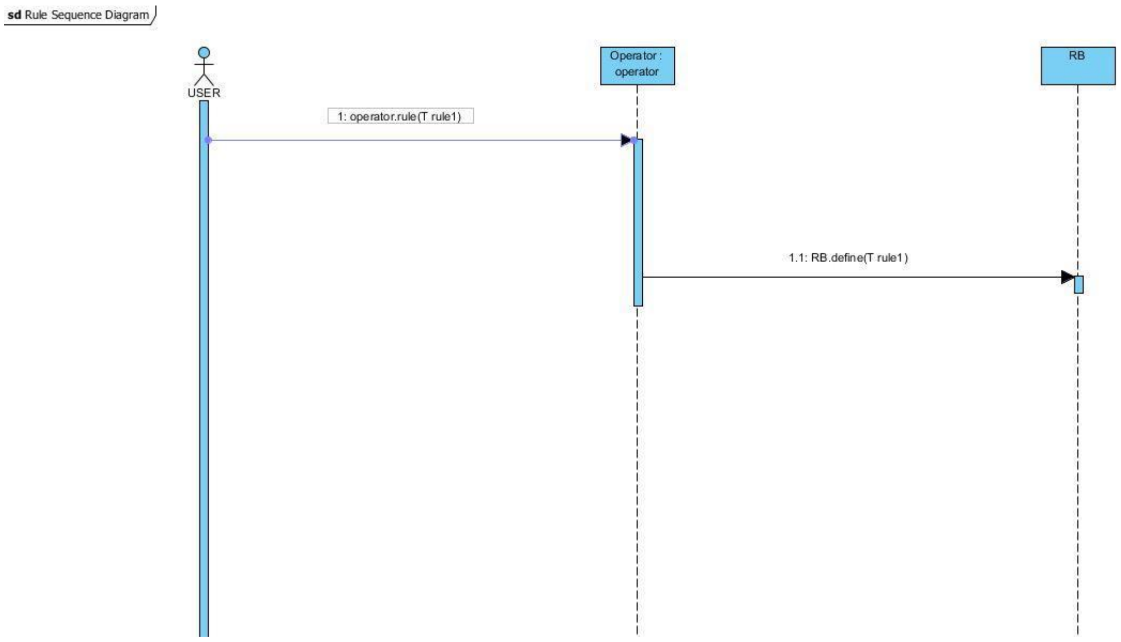

User
provide an sri path as the parameter
LOAD
OPER
load fact
load rule
define fact
KB
CONSOLE
RB
User
parameter: provide fact rule
FACT RULE
dump fact
if search finds rule, go into KB under fact and print to console
define rule
User
provide an sri path as the parameter
DUMP
dump rule
drop rule
User
INFERENCE
SEARCH
found
!found
drop fact
User
DROP

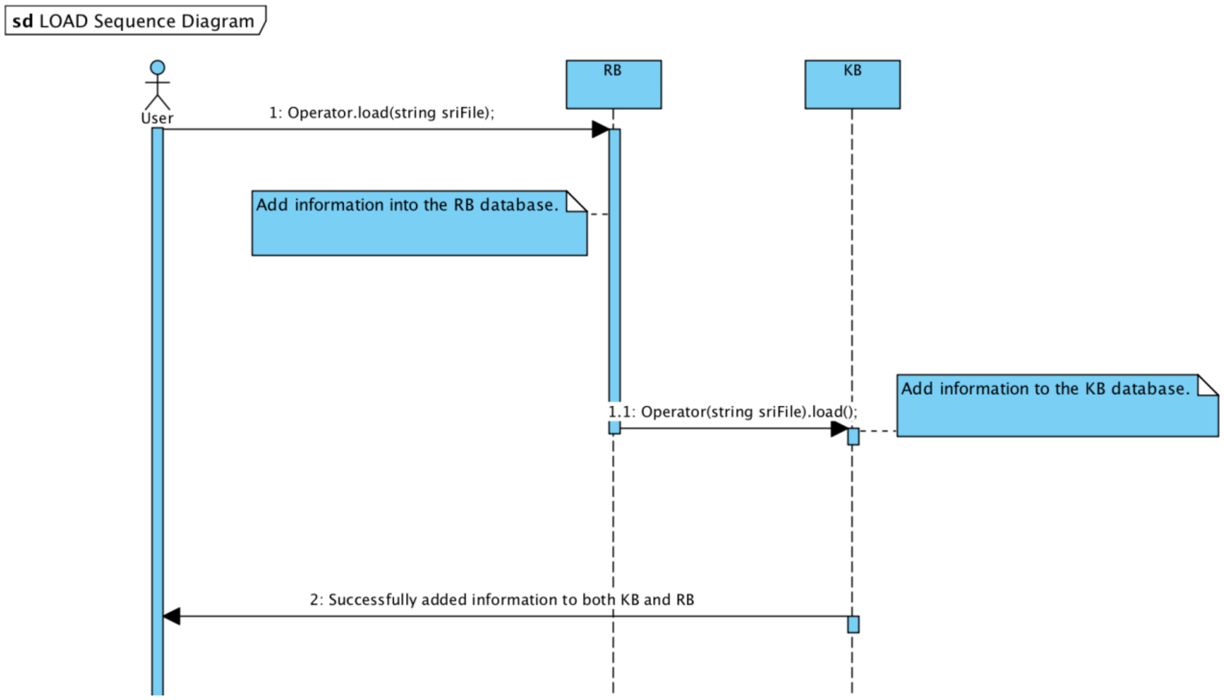

interaction drop
OPE TION
KB
RB
drop rule
drop fact


interaction inference
OPERATOR
RB
KB
if found()
: found
3 : found
!found()
4 : Message2


Actor
operator
RB
KB
1: 1. operator.dump(.sri)
1.1: RB.container
1.2: KB.container
sd exists
2: writetofile
3: writetofile
sd !exists
4:
5:


sd Fact Sequence Diagram
USER
Operator : operator
KB
1: operator.fact(T fact1)
1.1: KB.define(T fact)


sd Rule Sequence Diagram
USER
Operator : operator
RB
1: operator.rule(T rule1)
1.1: RB.define(T rule1)


sd LOAD Sequence Diagram
User
RB
KB
1: Operator.load(string sriFile);
Add information into the RB database.
1.1: Operator(string sriFile).load();
Add information to the KB database.
2: Successfully added information to both KB and RB