✯ Jarrett van den Bergh
Jennifer Fullerton
Kyle Oppenheim
Christopher Paz

SRI DESIGN DOCUMENT 2.0

Overview and Assumptions

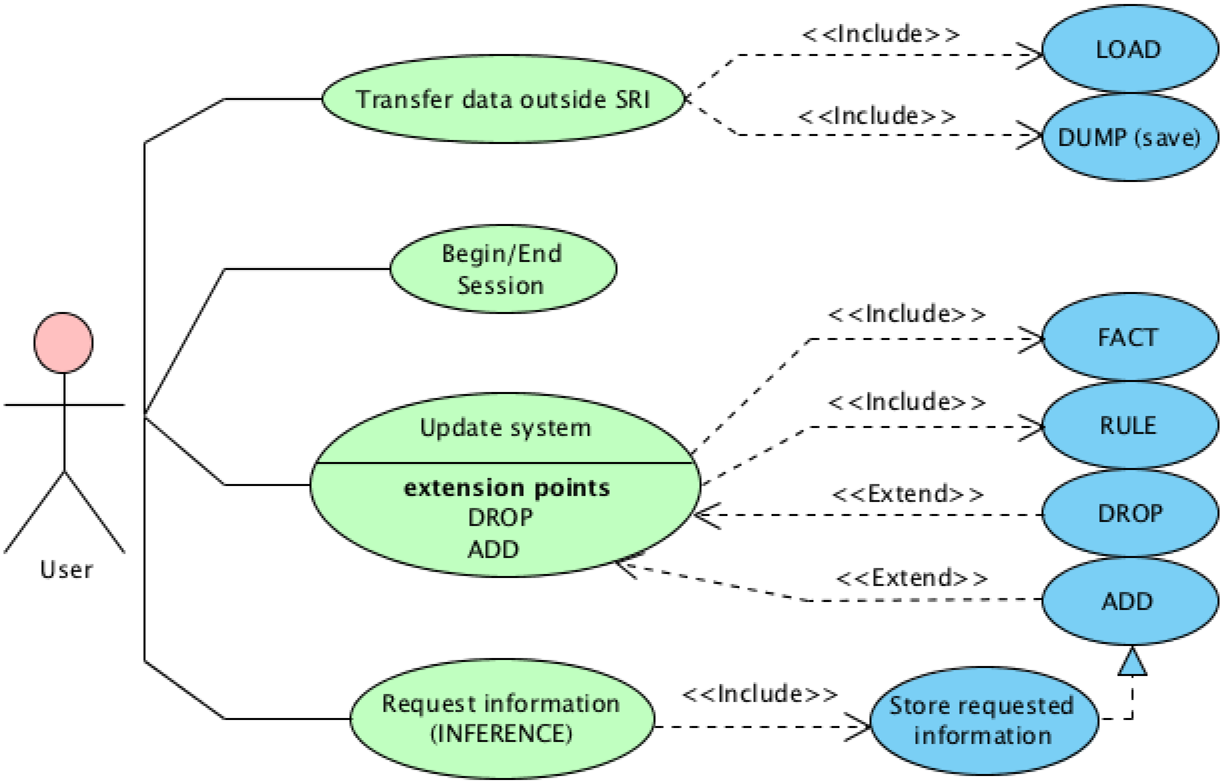
The main class simply loops through input and sends it to the SRI engine through the InterpretLine function. The program will not correct any misplaced spaces - there needs to be no spaces in each clause e.x. Father(A,B) - but SRI will usually throw an error on invalid input. The MakeFile has the following functions:

'make'	:	Compile the program normally (-std=gnu++11)
'make run'	:	Compile and run the program
'make test'	:	Compile the program with TestSRI.cpp instead of main.cpp
'make testrun'	:	Compile and run the program with TestSRI.cpp
'make clean'	:	Remove all executable and object files
'make clean_files'	:	Remove all .sri file
'make check'	:	Run valgrind on the already compiled program

We use the Catch library to do our unit testing, so we have included Catch.hpp.

On design decisions, we have the results of an AND inference pipe through from clause to clause. The alternative to this was to calculate each clause separately and then find the intersection between clauses. We believe our approach to be faster.

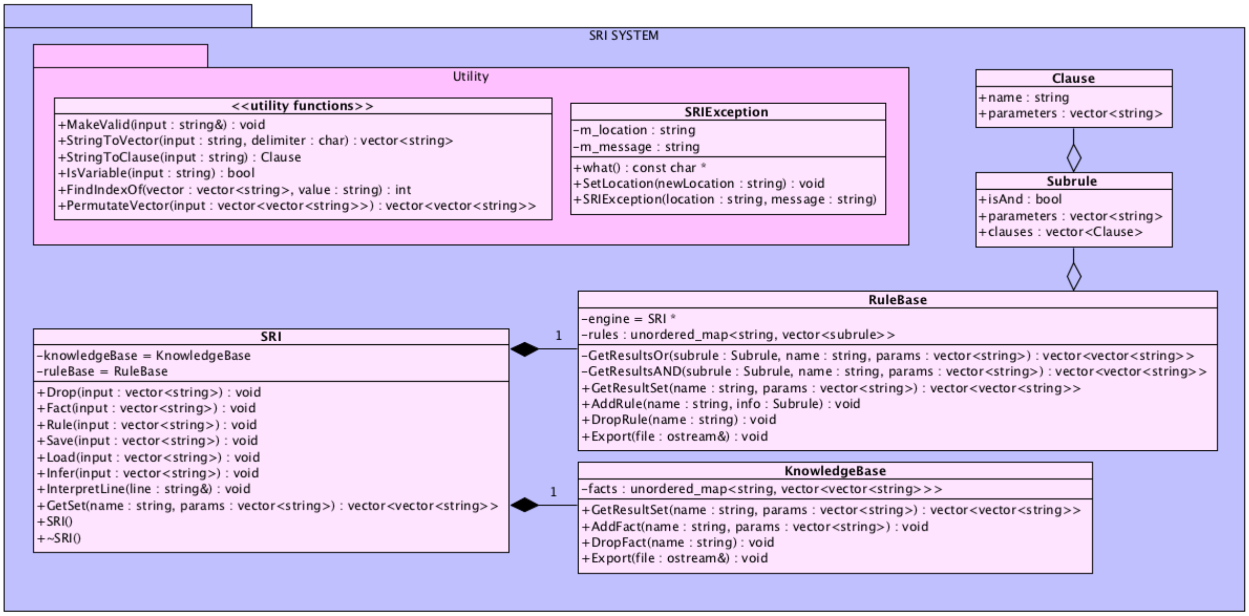
Use Case Diagram



The user will be able to call the following commands:

FACT (f)	: Updates system by adding fact.
RULE (r)	: Updates system by adding rule.
DROP (d)	: Updates system by dropping fact.
INFERENCE (i)	: Requests information on rule/fact, optionally stores the result as a new fact.
LOAD (l)	: Transfers data from SRI into file
DUMP (s)	: Transfers data from file into SRI
EXIT (x)	: Ends session

Class Diagram


SRI SYSTEM
Utility
<<utility functions>>
+MakeValid(input : string&) : void
+StringToVector(input : string, delimiter : char) : vector<string>
+StringToClause(input : string) : Clause
+IsVariable(input : string) : bool
+FindIndexOf(vector : vector<string>, value : string) : int
+PermutateVector(input : vector<vector<string>>) : vector<vector<string>>
SRIException
-m_location : string
-m_message : string
+what() : const char *
+SetLocation(newLocation : string) : void
+SRIException(location : string, message : string)
Clause
+name : string
+parameters : vector<string>
Subrule
+isAnd : bool
+parameters : vector<string>
+clauses : vector<Clause>
RuleBase
-engine = SRI *
-rules : unordered_map<string, vector<subrule>>
-GetResultsOr(subrule : Subrule, name : string, params : vector<string>) : vector<vector<string>>
-GetResultsAND(subrule : Subrule, name : string, params : vector<string>) : vector<vector<string>>
+GetResultSet(name : string, params : vector<string>) : vector<vector<string>>
+AddRule(name : string, info : Subrule) : void
+DropRule(name : string) : void
+Export(file : ostream&) : void
SRI
-knowledgeBase = KnowledgeBase
-ruleBase = RuleBase
+Drop(input : vector<string>) : void
+Fact(input : vector<string>) : void
+Rule(input : vector<string>) : void
+Save(input : vector<string>) : void
+Load(input : vector<string>) : void
+Infer(input : vector<string>) : void
+InterpretLine(line : string&) : void
+GetSet(name : string, params : vector<string>) : vector<vector<string>>
+SRI()
+~SRI()
1
1
KnowledgeBase
-facts : unordered_map<string, vector<vector<string>>>
+GetResultSet(name : string, params : vector<string>) : vector<vector<string>>
+AddFact(name : string, params : vector<string>) : void
+DropFact(name : string) : void
+Export(file : ostream&) : void

SRI

- Holds reference to our knowledge and rule base.
- Is the only class to directly interface with the client.
- Handles input through InterpretLine, then calls helper functions.
- GetSet is public so that RuleBase may recursively infer on other rules/facts.

KnowledgeBase

- Holds all facts as a dictionary of name → list (fact parameters).

RuleBase

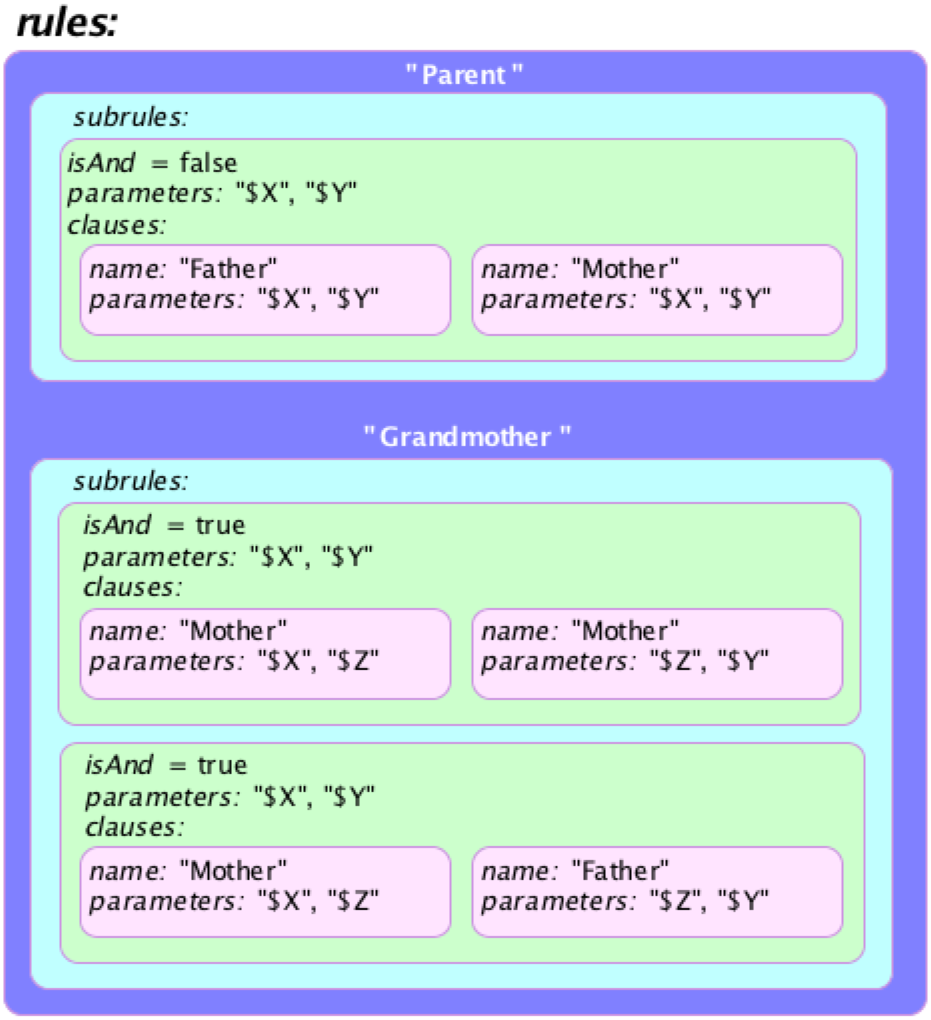
- Holds forward declared reference to SRI for recursion.
- Holds all rules as a dictionary of name → list (subrules).
- Uses Subrule and Clause as data-only structs.

Pseudo Object Diagram

Input

```
RULE Parent($X,$Y):- OR Father($X,$Y) Mother($X,$Y)
RULE GrandMother($X,$Y):- AND Mother($X,$Z) Mother($Z,$Y)
RULE GrandMother($X,$Y):- AND Mother($X,$Z) Father($Z,$Y)
```

RuleBase



rules:

"Parent"

subrules:

isAnd = false
parameters: "$X", "$Y"
clauses:

name: "Father"
parameters: "$X", "$Y"

name: "Mother"
parameters: "$X", "$Y"

"Grandmother"

subrules:

isAnd = true
parameters: "$X", "$Y"
clauses:

name: "Mother"
parameters: "$X", "$Z"

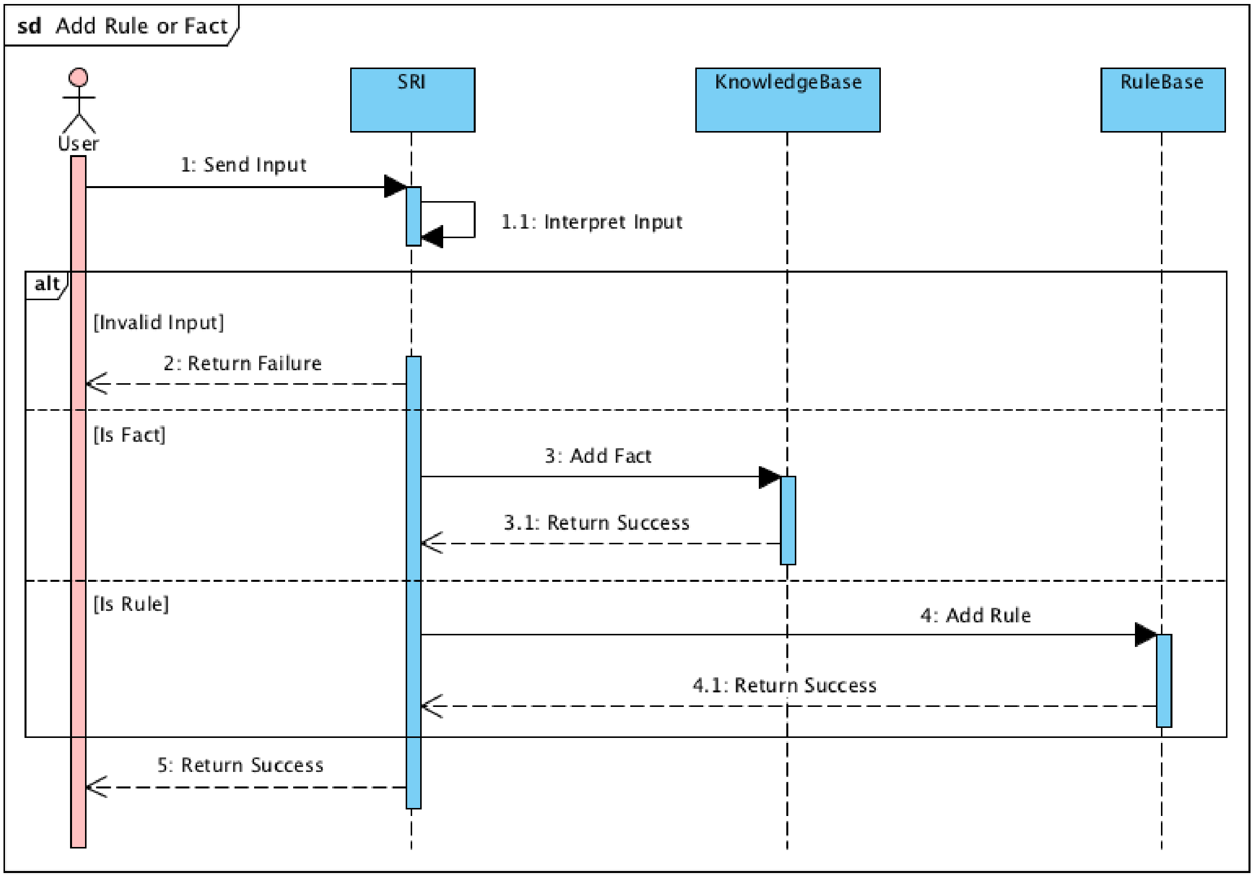
name: "Mother"
parameters: "$Z", "$Y"

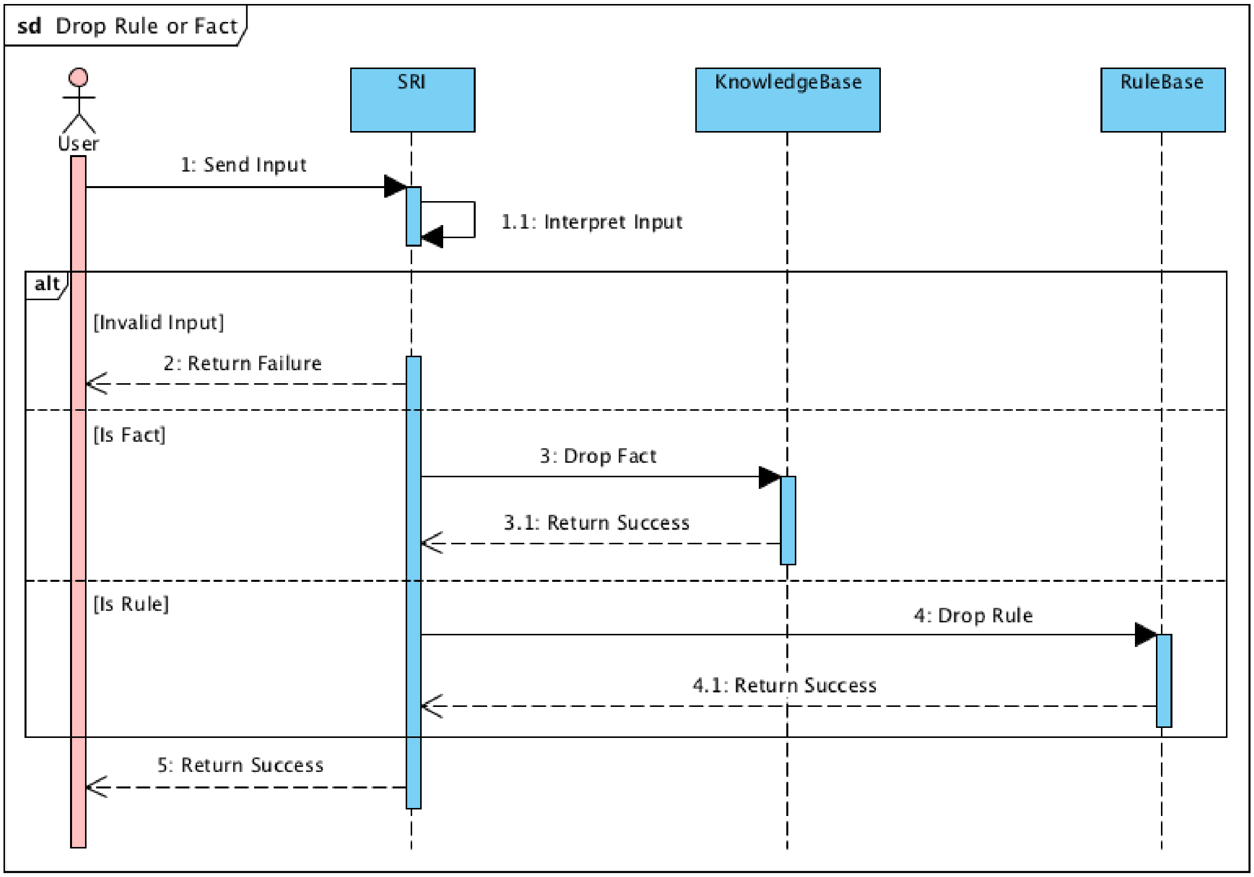
isAnd = true
parameters: "$X", "$Y"
clauses:

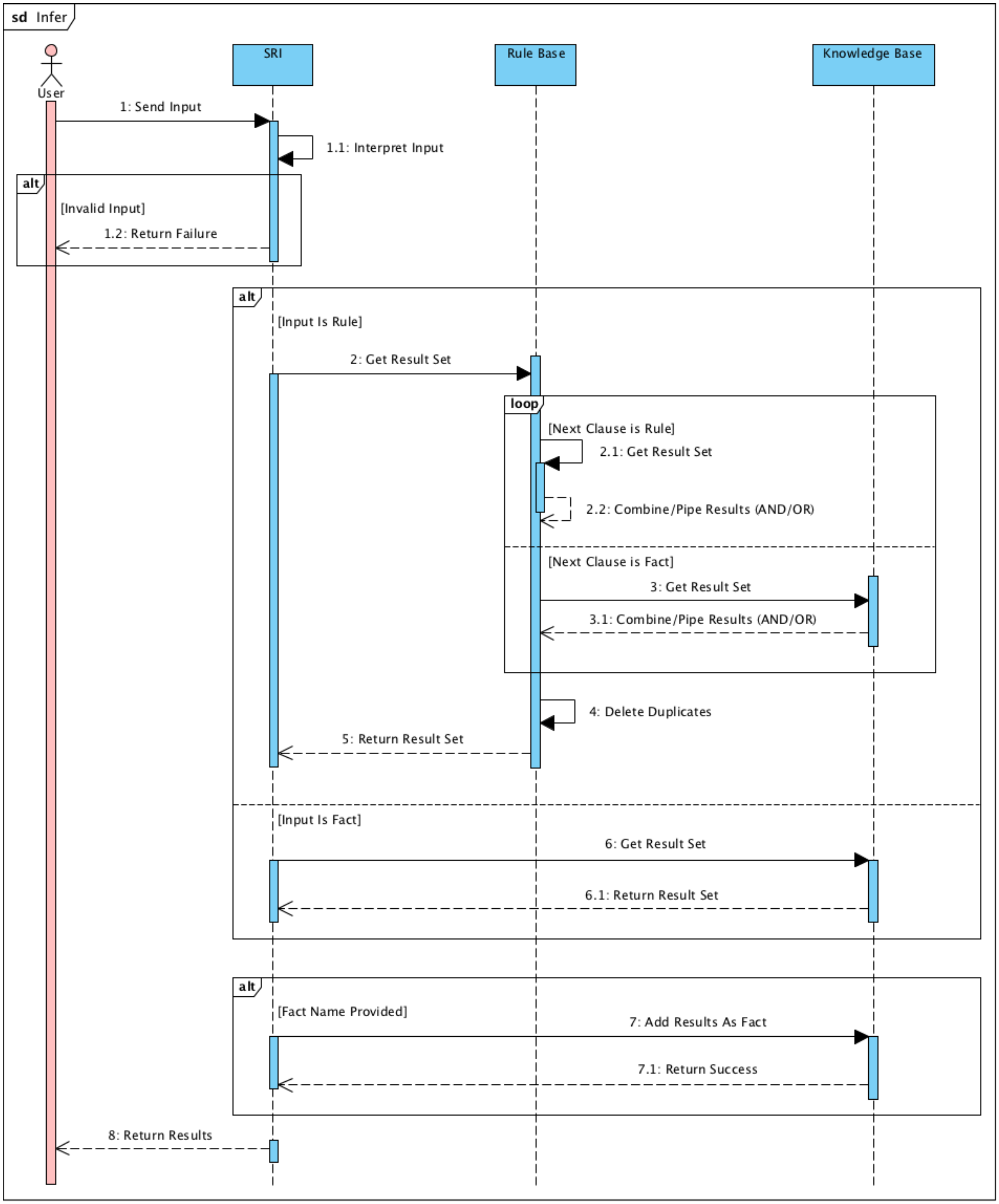
name: "Mother"
parameters: "$X", "$Z"

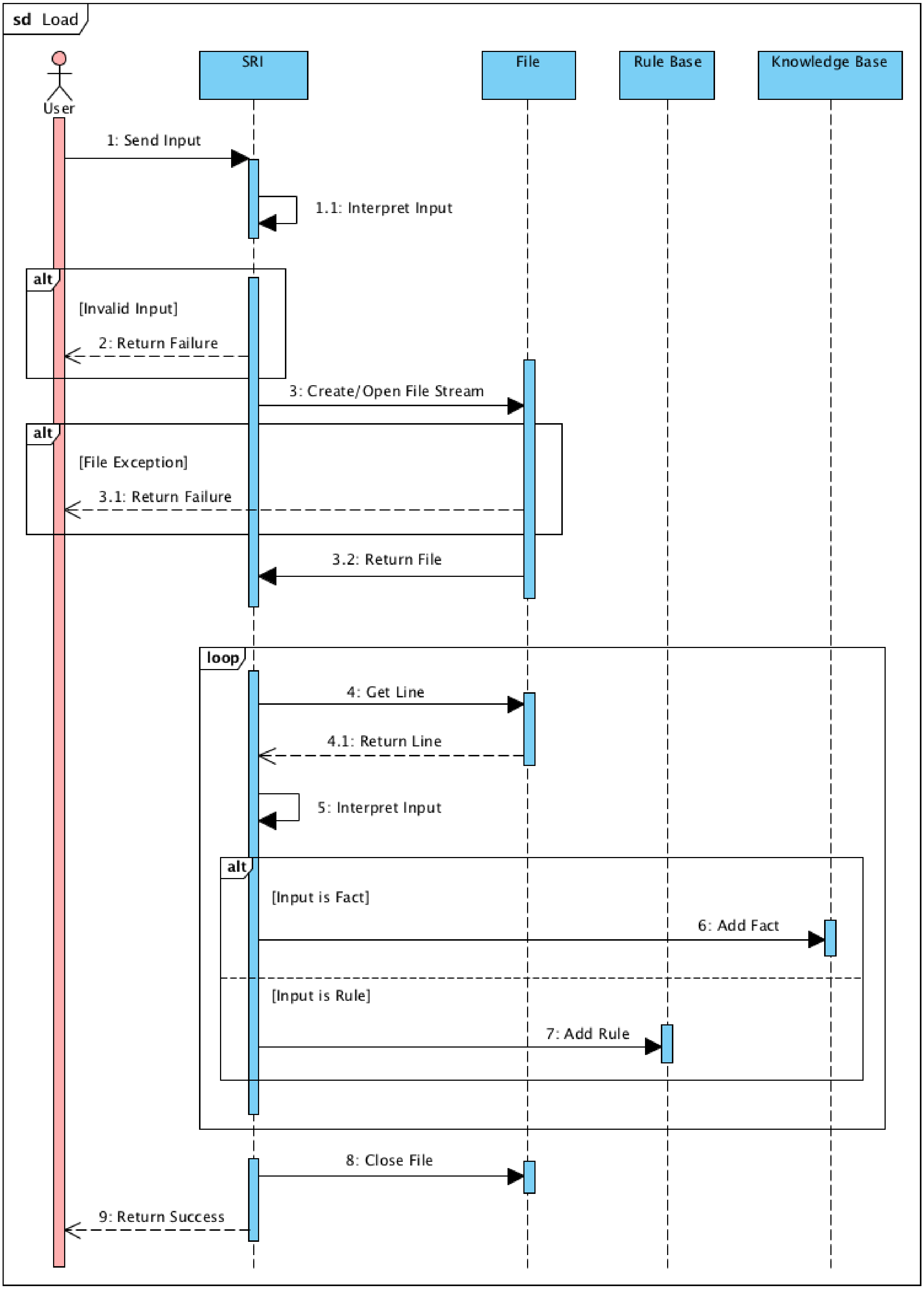
name: "Father"
parameters: "$Z", "$Y"

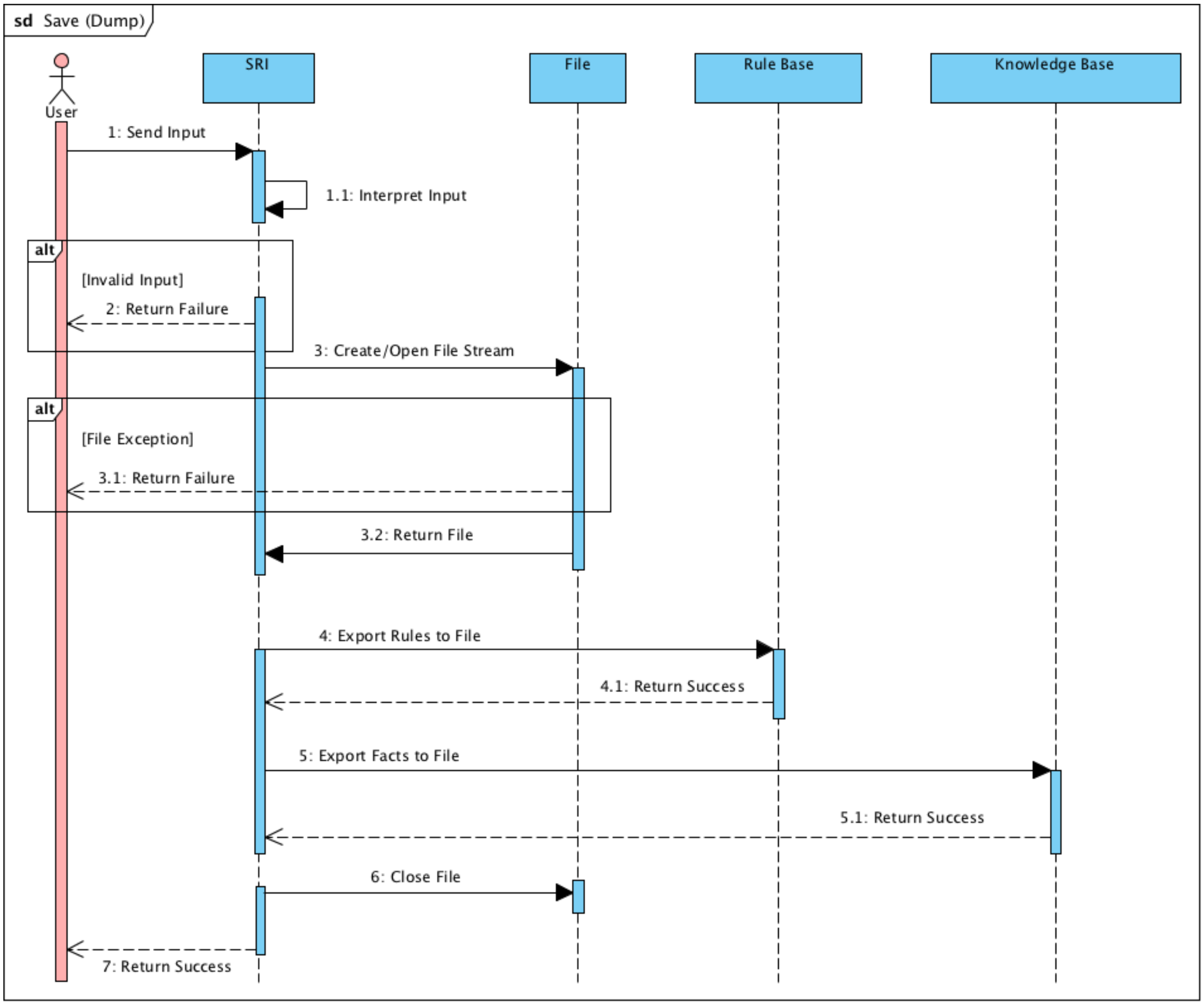
Sequence Diagrams



sd Drop Rule or Fact
User
SRI
KnowledgeBase
RuleBase
1: Send Input
1.1: Interpret Input
alt
[Invalid Input]
2: Return Failure
[Is Fact]
3: Drop Fact
3.1: Return Success
[Is Rule]
4: Drop Rule
4.1: Return Success
5: Return Success


sd Infer
User
SRI
Rule Base
Knowledge Base
1: Send Input
1.1: Interpret Input
alt
[Invalid Input]
1.2: Return Failure
alt
[Input Is Rule]
2: Get Result Set
loop
[Next Clause is Rule]
2.1: Get Result Set
2.2: Combine/Pipe Results (AND/OR)
[Next Clause is Fact]
3: Get Result Set
3.1: Combine/Pipe Results (AND/OR)
4: Delete Duplicates
5: Return Result Set
[Input Is Fact]
6: Get Result Set
6.1: Return Result Set
alt
[Fact Name Provided]
7: Add Results As Fact
7.1: Return Success
8: Return Results


sd Load
User
SRI
File
Rule Base
Knowledge Base
1: Send Input
1.1: Interpret Input
alt
[Invalid Input]
2: Return Failure
3: Create/Open File Stream
alt
[File Exception]
3.1: Return Failure
3.2: Return File
loop
4: Get Line
4.1: Return Line
5: Interpret Input
alt
[Input is Fact]
6: Add Fact
[Input is Rule]
7: Add Rule
8: Close File
9: Return Success


sd Save (Dump)
User
SRI
File
Rule Base
Knowledge Base
1: Send Input
1.1: Interpret Input
alt
[Invalid Input]
2: Return Failure
3: Create/Open File Stream
alt
[File Exception]
3.1: Return Failure
3.2: Return File
4: Export Rules to File
4.1: Return Success
5: Export Facts to File
5.1: Return Success
6: Close File
7: Return Success