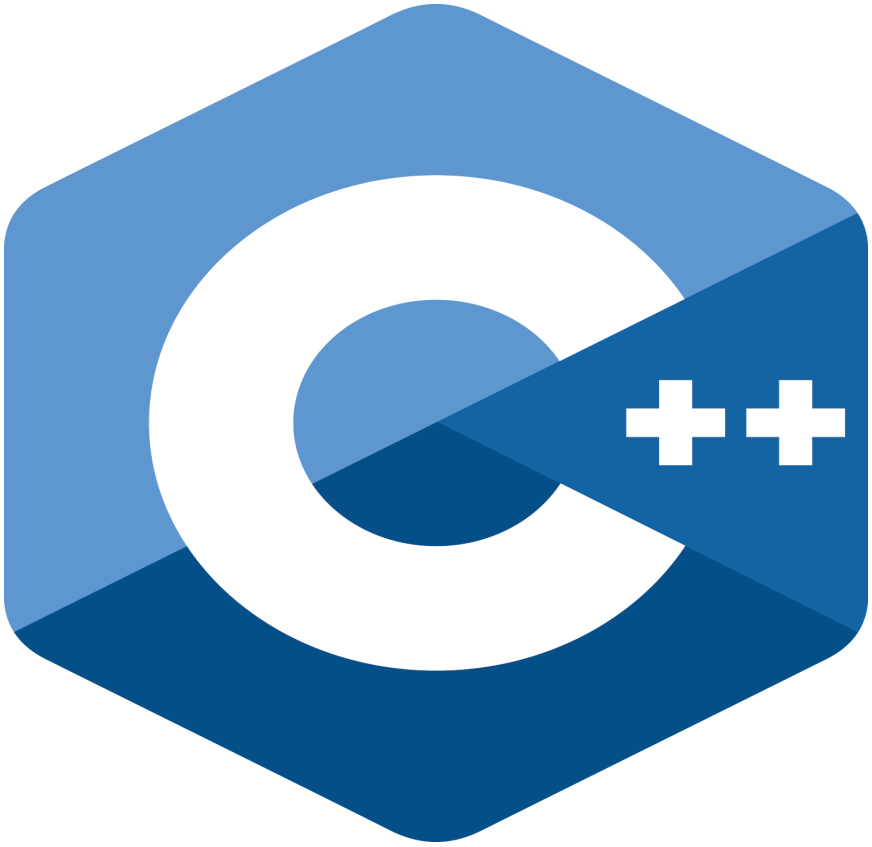
CMPS 109 Winter 2017: Quarter Project Phase 3



Ahmed Elshaarany

Marian Vladoi

Nazanin Tafrishi

Thomas Shum

03.10.2017

PHASE 3 CHANGES WITH RESPECT TO PHASE 2:

1. **Sequence Diagram changes:**
   The Inference Sequence Diagram on page 7 of this document was modified to indicate that we create a new thread for each predicate of a rule.

2. **Or Rule Inference:**
   a. We created a new class called predicateThread that inherits from the Thread class

   b. Lines 17-45 in predicateThread.cpp show the predicateThread threadMainBody implementation for Or rule inference locks the master table for each thread. The master table contains all the entries that the predicates return. We used a print mutex to lock printing to the screen so that the thread numbers printing would be displayed correctly.

   c. Lines 558-611 in sri_engine.cpp show that for each valid predicate, we create a new thread and it to a thread manager. The thread manager then fires all threads at the same time and wait for all of them to finish.

   d. We used a print mutex for printing the thread numbers so that it would be displayed correctly.

3. **And Rule Inferenece:**
   a. We created a new class called andThread that inherits from the Thread class

   b. The code that each thread executes is not stored in threadMainBody due to its dependence on private functions and members in sri_engine. Instead, each thread executes a static function in sri_engine called andThredFunction() [sri_engine.cpp:638-684]

   c. To make dealing with creating threads easier, a static method called createAndThread [sri_engine.cpp:686-707] helps to make setting up the thread easier by wrapping up all the parameters in a struct allocated on the heap and passing it to the thread object.

   d. Each thread represents the expansion of an entry from one subtable, which could result in 50+ threads spawning altogether depending on how many entries

each predicate expands to and such.

ASSUMPTIONS (BASED ON PHASE 2)

1. Facts and rules can have an arbitrary number of parameters
2. Two parameter strings with the same content refer to the same entity (e.g. Father(John, Mary); Mother (Mary, Jane) -- Mary is the same in both facts)
3. Rules have an arbitrary number of predicates
4. Rules can take other rules as predicates
5. Rules have only two logical operators: AND and OR.
6. Multiple instances of a fact can have different number of parameters
7. Multiple instances of a rule have the same number of parameters.
8. No Rule's parameters can have
9. A fact and a rule cannot share the same name

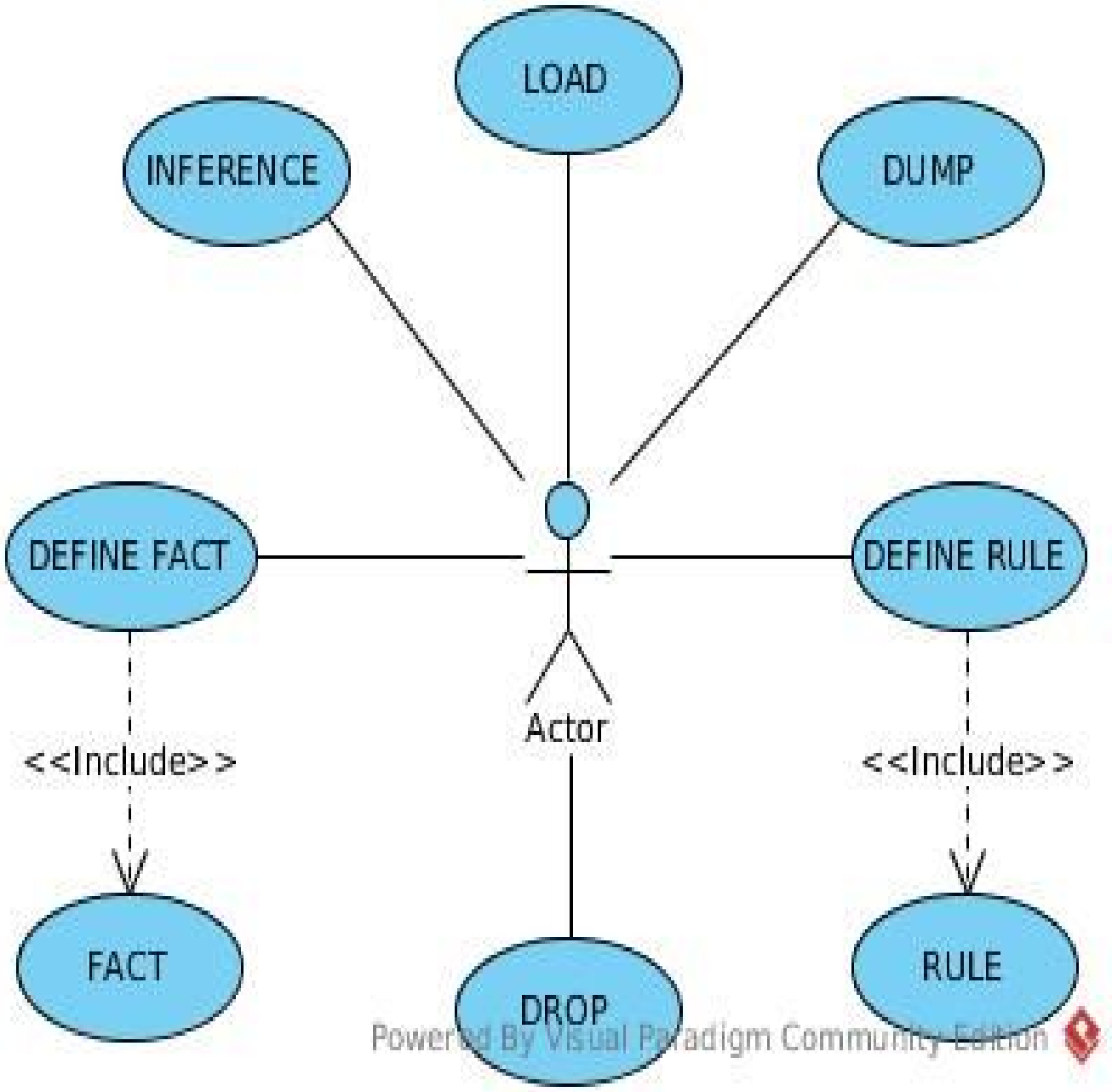
Using the User Interface (BASED ON PHASE 2)

- To Start the user interface, run the main program
- To exit the user interface, enter "exit"
- Input commands as defined in the project writing and
- As an example, you can use the provided family_relation.sri file
- Note: please follow the same syntax as the provided family_relation.sri to avoid program crashes

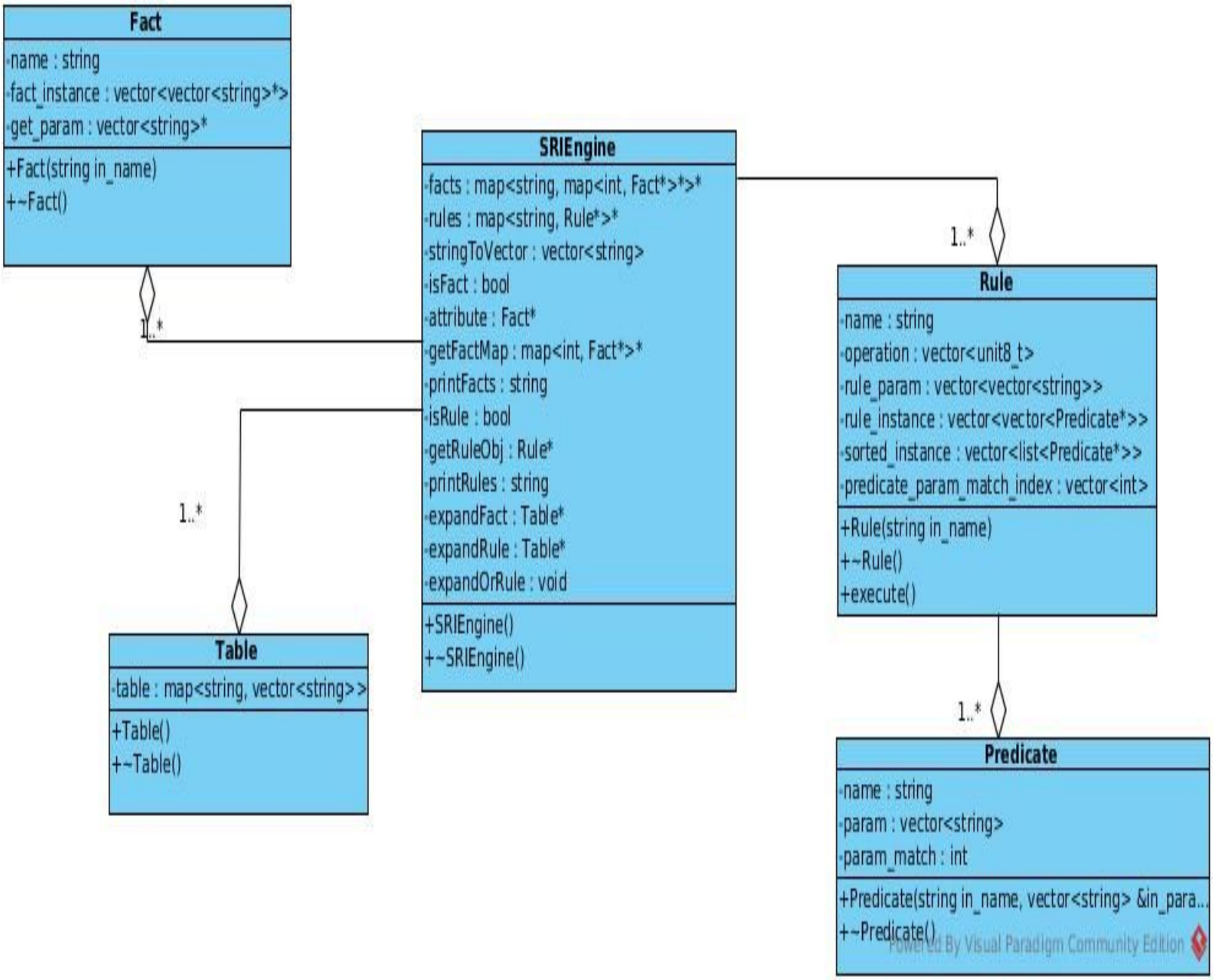
Functionalities Supported (BASED ON PHASE 2)

- OR/AND with general query parameters (ex: INFERENCE Parent($X,$Y))
- OR/AND with specific query parameters (ex: INFERENCE Parent($X,Albert))
- OR/AND with all specific query parameters will return "True" if the INFERENCE does exist, otherwise it will return "False"
- LOAD from a file
- DUMP into a file
- Add a FACT/RULE
- Drop a FACT/RULE
- Saving an inference as a new Fact

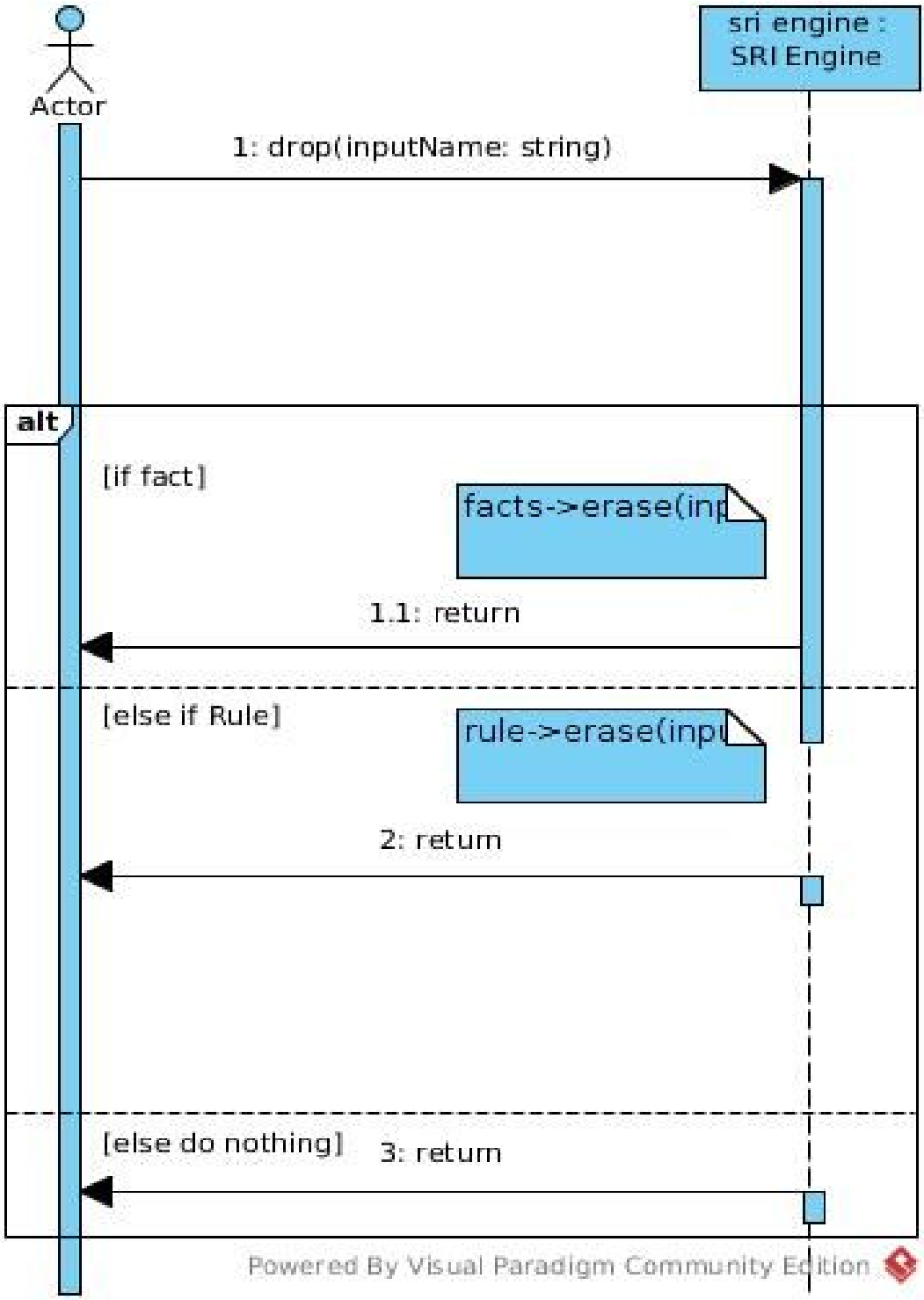
Use case


LOAD
INFERENCE
DUMP
DEFINE FACT
DEFINE RULE
Actor
<<Include>>
<<Include>>
FACT
DROP
RULE
Powered By Visual Paradigm Community Edition

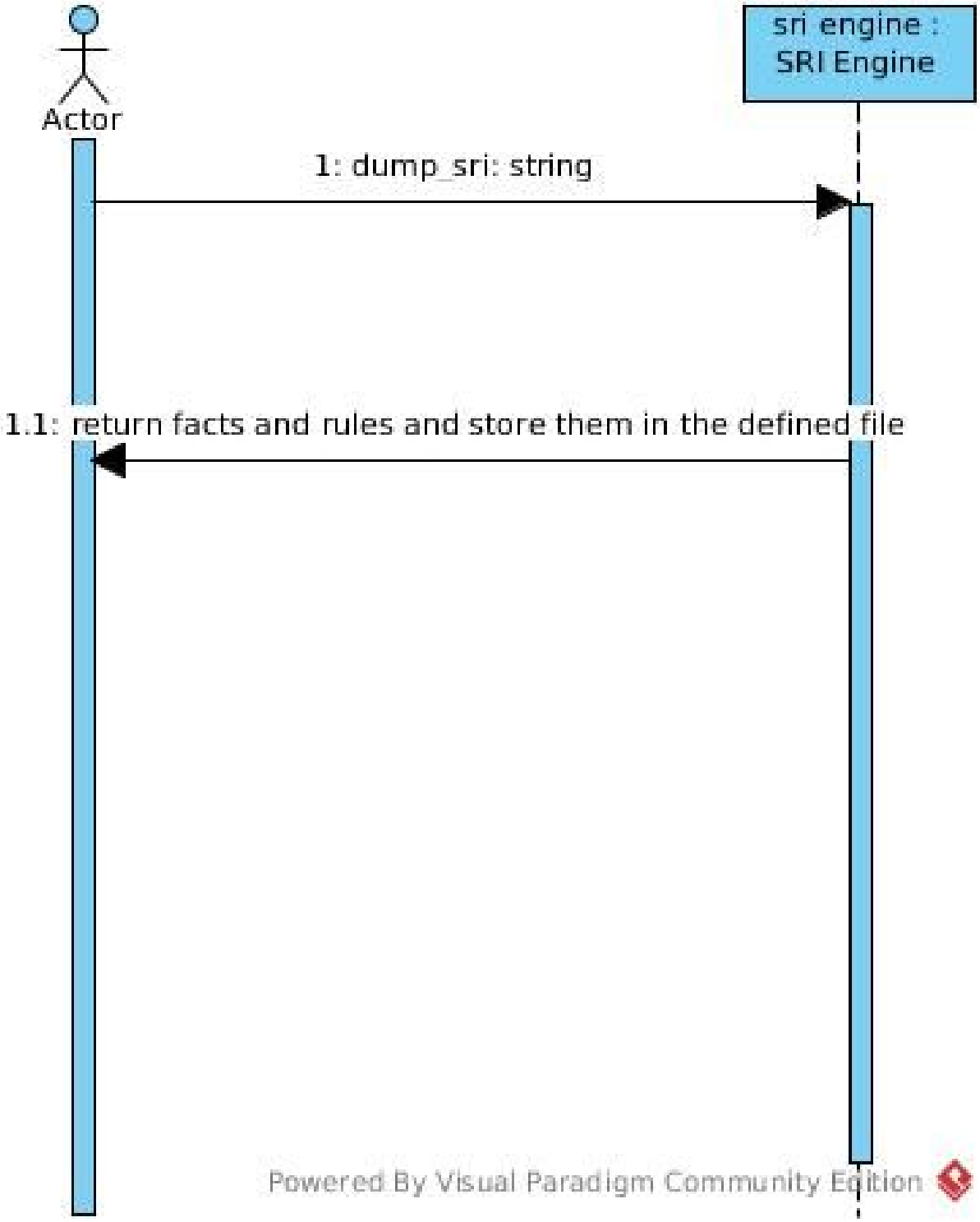
Class Diagram



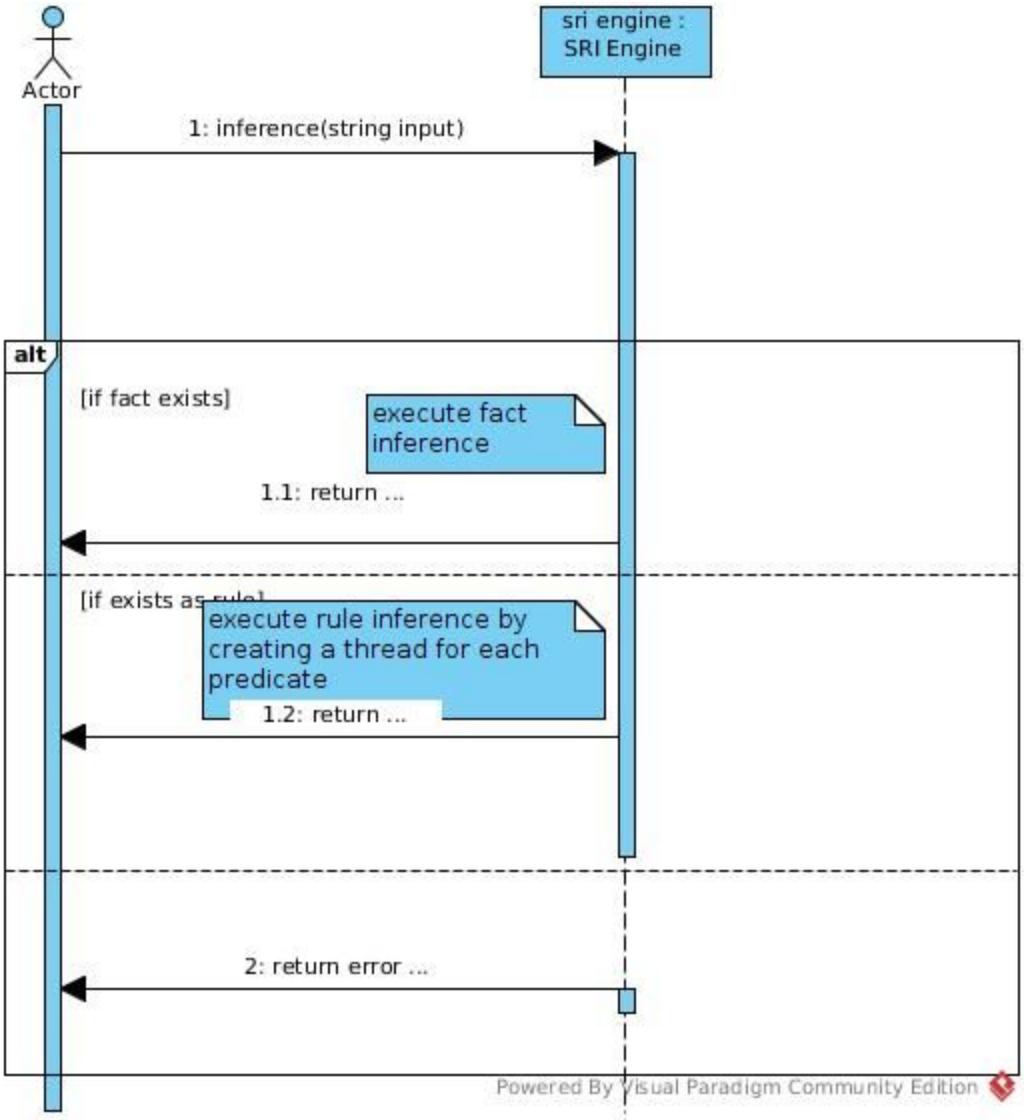
Sequence - Drop Fact/Rule



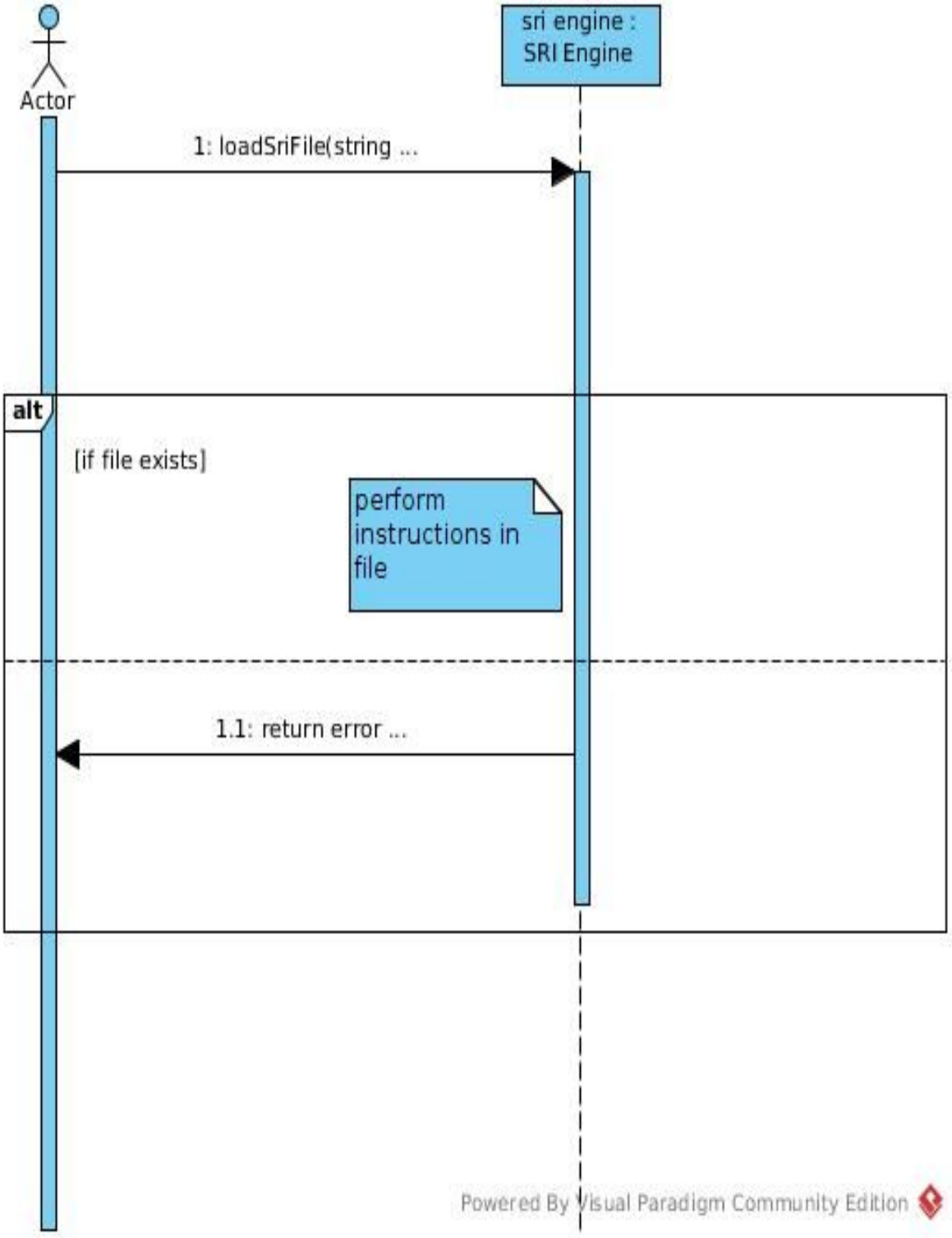
Sequence - Dump



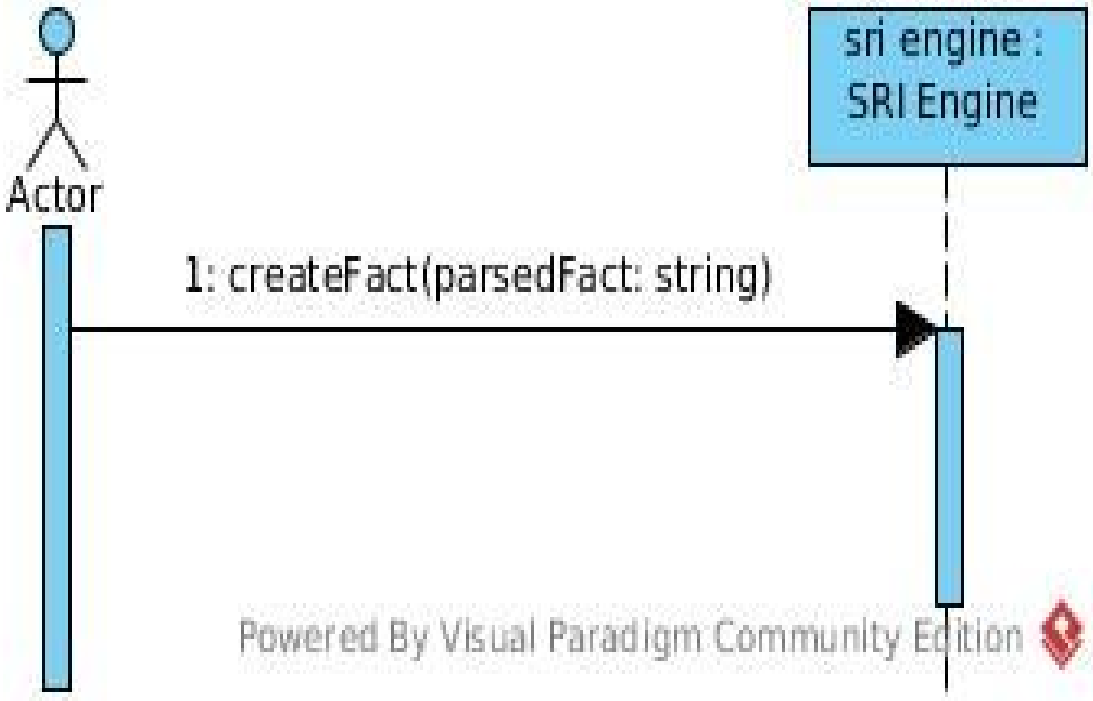
Sequence - Inference



Sequence - Load



Sequence - Define FACT



Sequence - Define RULE