COMP30040 Third Year Project

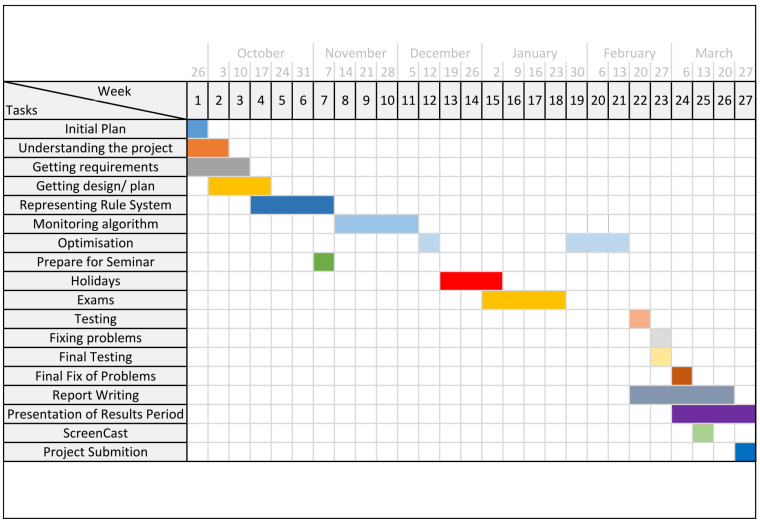
Runtime Verification Tool using RuleR Language

Mantas Daraciunas

The University Of Manchester October 16, 2016

Contents

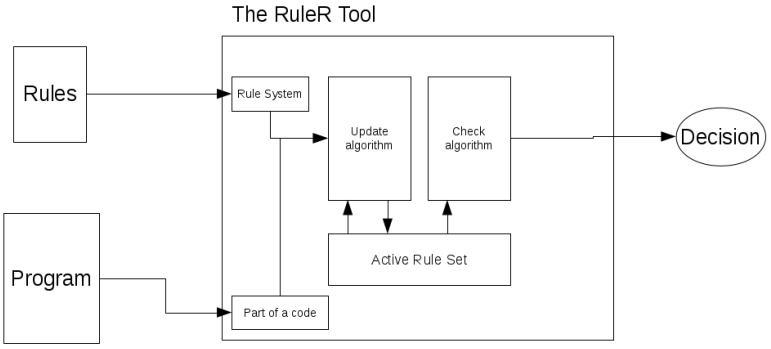
1 Timeline


October
November
December
January
February
March
26 3 10 17 24 31 7 14 21 28 5 12 19 26 2 9 16 23 30 6 13 20 27 6 13 20 27
Week
Tasks
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Initial Plan
Understanding the project
Getting requirements
Getting design/ plan
Representing Rule System
Monitoring algorithm
Optimisation
Prepare for Seminar
Holidays
Exams
Testing
Fixing problems
Final Testing
Final Fix of Problems
Report Writing
Presentation of Results Period
ScreenCast
Project Submition

2 Requirements

To create a tool which is able to monitor the events in runtime and decide whether the sequence of the events has happend according to the rules provided.

3 Descriptions



Rule System Predefined set of rules which define the expected flow of the program. Rule System should contain one or more rules. Each rule has to have list of the events with optional conditions which will lead each event to some action another rule or verdict (success or failure). A Rule is of the form

$$ModifierRuleName(Parameters)\{List[RulePart]\}$$

where
Modifier = always or state or step
RulePart = EventName(Parameters) [Optional Conditions] -¿ List[Action]
Action = RuleName(Parameters) or Ok or Fail

The **Modifiers** of the rule can be as following:
always = always keep this rule activation
state = remove this rule activation if it fires
step = only keep this rule activation for one step

A **Rule System** is a tuple ¡ R, S, F, A ¿
where R is a set of Rules,
S, F and A are sets of starting, forbidden and asserted rule names.
S must contain at least one rule name.
F and A may be empty

Active Rule Set is a set of the current activated rules in the monitored trace. Rule Activations are represented as rule name with the parameter and value. Each rule could have more than one Rule Activation in Active Rule Set

Update stage the part of runtime verification tool which updates rules in Active Rule Set. It takes piece of code and rule from Rule System and replaces, deletes, introduce or keep existing rules in Active Rule Set.

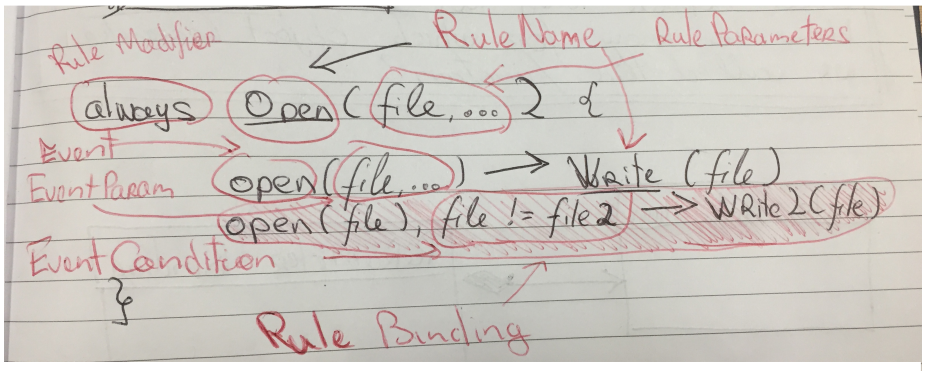
The update algorithm takes an event and naively does the following

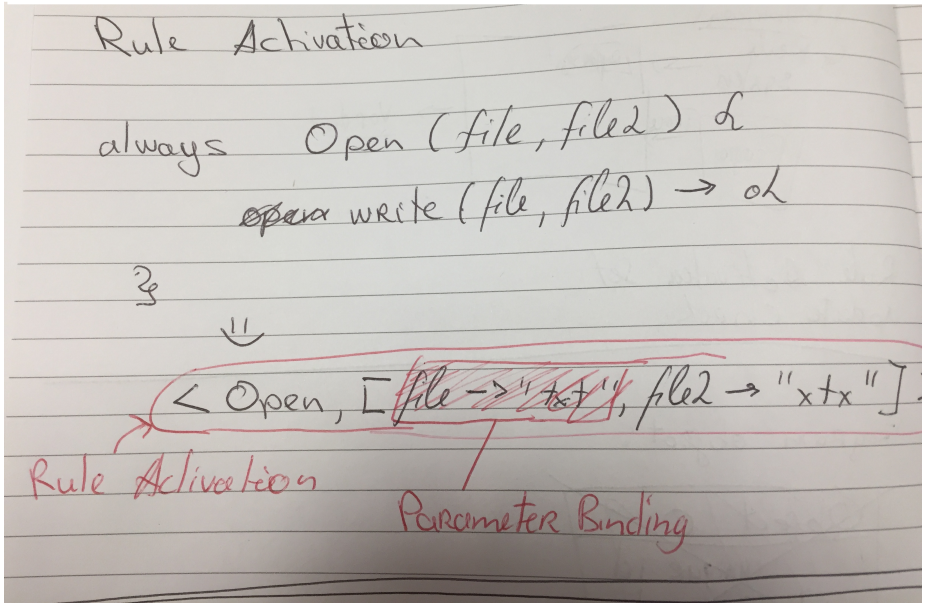
for each rule activation ¡ruleName, binding¿ in the Active Rule Set **do**
  for foreach RuleBody in ruleName **do**
    if the event matches the left of the RuleBody **then**
      write the binding
      update the binding and use it to add the right side
      if the modifier of RuleBody is state **then**
        remove the rule activation
      end if
      if the modifier of step **then**
        then remove the rule activation
      end if
    end if
  end for
end for

The stuff to do with bindings we should discuss at a later date, when you need to implement the update algorithm.

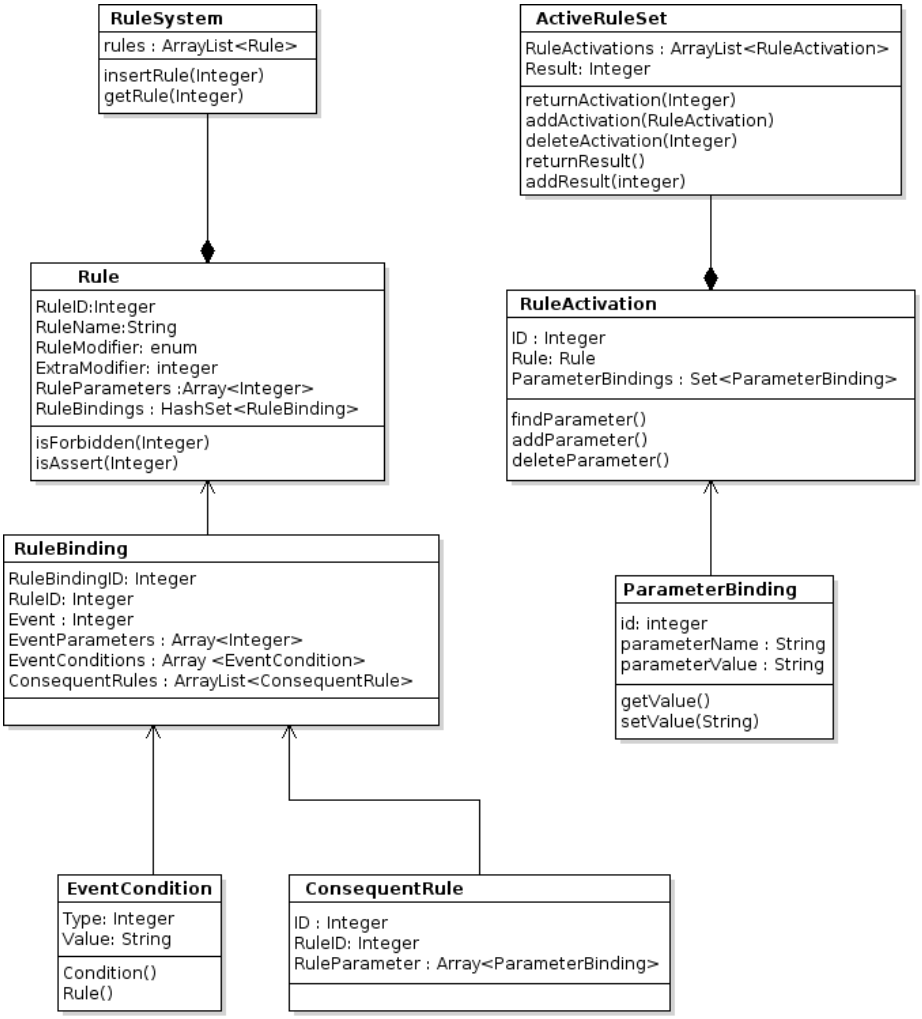
Decision stage - the part of runtime verification tool which takes Results of each rule from Active Rule Set and generates a decision on to the screen. The decision algorithm produces a verdict after each event. It uses the Active Rule Set and the sets A and F to do this.
Approximately it decides as follow, where ARS is the Active Rule Set:

if Fail is in ARS **then**
  decide Failure
else
  if the ARS is empty **then**
    decide Success
  end if
else
  if we have not just fired a rule in A **then**
    decide Failure
  end if
else
  if there is a rule name in ARS that is also in F **then**
    decide WeakFailure
  end if
else
  otherwise decide WeakSuccess
end if


Rule Modifier
Rule Name
Rule Parameters
always
Open (file, ...) {
Event
Event Param
open (file, ...) → Write (file)
open (file), file != file2 → WRite2 (file)
Event Condition
}
Rule Binding


Rule Activation
always Open (file, file2) {
write (file, file2) → o{
}
< Open, [file → "txt", file2 → "xtx"] >
Rule Activation
Parameter Binding

4 Design


RuleSystem
rules : ArrayList<Rule>
insertRule(Integer)
getRule(Integer)
ActiveRuleSet
RuleActivations : ArrayList<RuleActivation>
Result: Integer
returnActivation(Integer)
addActivation(RuleActivation)
deleteActivation(Integer)
returnResult()
addResult(integer)
Rule
RuleID:Integer
RuleName:String
RuleModifier: enum
ExtraModifier: integer
RuleParameters :Array<Integer>
RuleBindings : HashSet<RuleBinding>
isForbidden(Integer)
isAssert(Integer)
RuleActivation
ID : Integer
Rule: Rule
ParameterBindings : Set<ParameterBinding>
findParameter()
addParameter()
deleteParameter()
RuleBinding
RuleBindingID: Integer
RuleID: Integer
Event : Integer
EventParameters : Array<Integer>
EventConditions : Array <EventCondition>
ConsequentRules : ArrayList<ConsequentRule>
ParameterBinding
id: integer
parameterName : String
parameterValue : String
getValue()
setValue(String)
EventCondition
Type: Integer
Value: String
Condition()
Rule()
ConsequentRule
ID : Integer
RuleID: Integer
RuleParameter : Array<ParameterBinding>