

---

COMP30040 Third Year Project

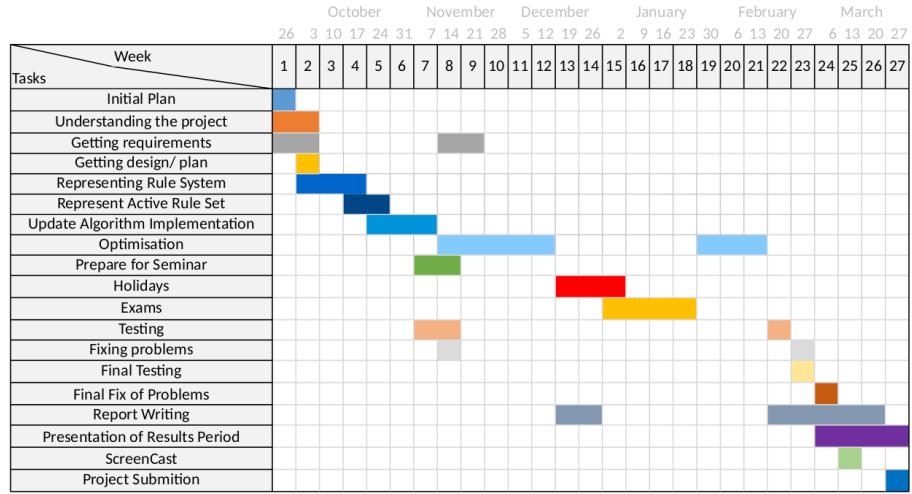
# Runtime Verification Tool using RuleR Language

Mantas Daraciunas

## Contents

<b>1</b>	<b>Timeline</b>	<b>3</b>
<b>2</b>	<b>Requirements</b>	<b>4</b>
<b>3</b>	<b>Descriptions</b>	<b>5</b>
<b>4</b>	<b>Design</b>	<b>8</b>

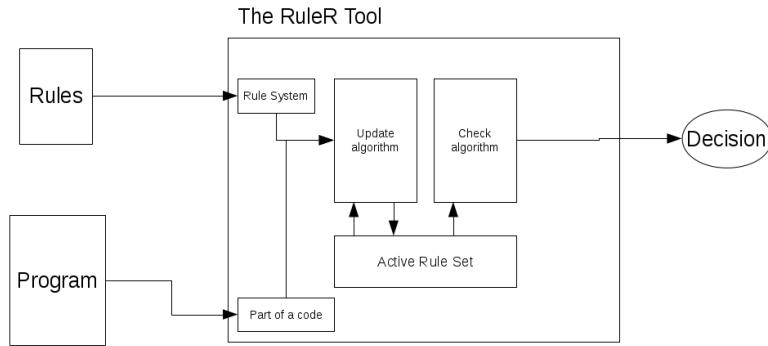
## 1 Timeline



## 2 Requirements

To create a tool which is able to monitor the events in runtime and decide whether the sequence of the events has happened 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

$$\text{Modifier} \text{Rule Name}(\text{Parameters})\{\text{List}[\text{RulePart}]\}$$

where

Modifier = always or state or step

RulePart = EventName(Parameters) [Optional Conditions] -i 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  $\langle R, S, F, A \rangle$

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.

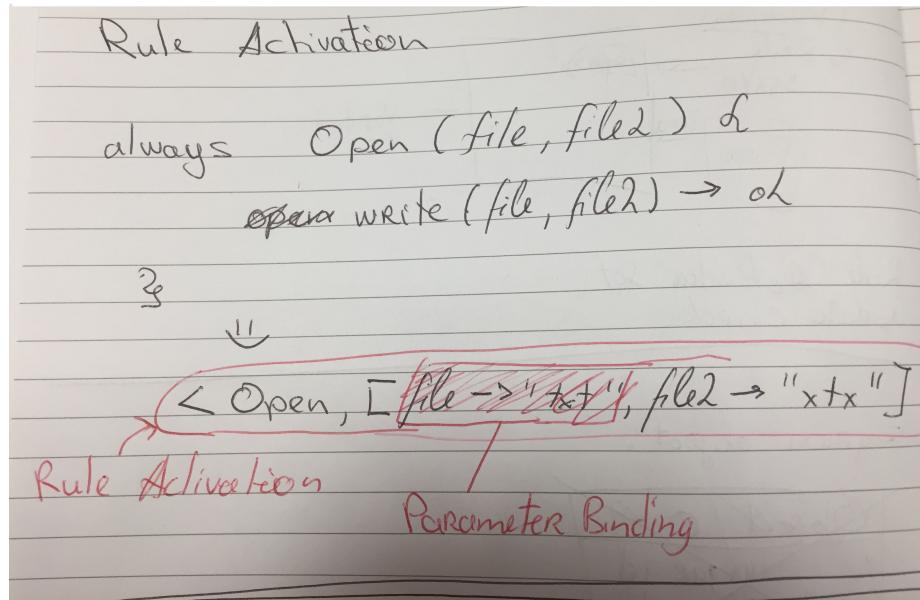
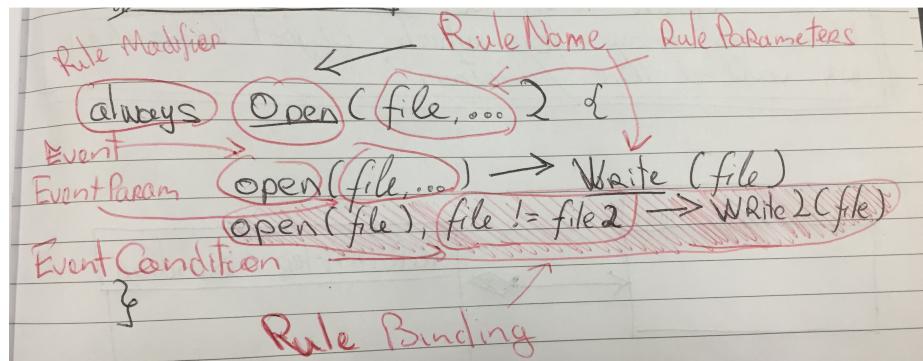
**Decision stage** - the part of runtime verification tool which takes Results of each rule from Active Rule Set and generates a decision on to the screen. The decision algorithm produces a verdict after each event. It uses the Active Rule Set and the sets A and F to do this.

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

```



## 4 Design

