Bridging the Gap between Requirements Modeling and Behavior-driven Development, Supplementary Materials

Mauricio Alferez*, Fabrizio Pastore*, Mehrdad Sabetzadeh*, Lionel C. Briand†, Jean-Richard Riccardi§

*SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
†School of Engineering and Computer Science, University of Ottawa, Canada
§Clearstream Services SA, Luxembourg

Email: {alferez, pastore, sabetzadeh, briand}@svv.lu, jean-richard.riccardi@clearstream.com

I. EXCERPT OF THE UML METAMODEL FOR ADs

![Excerpt of the UML metamodel for Activity Diagrams](image)

II. SELECTION OF CONCURRENT NODES

![Selection of concurrent nodes](image)

AGAC must take into account concurrent nodes. Therefore, the nodes in a path should be either non-concurrent or concurrent nodes satisfying precedence relationships between them. The precedence relation, denoted “≺”, is defined over a set of nodes N in a test model, by the application of three rules:

Rule 1: If a node \( n_i \in N \) precedes a ParallelStart node \( p \) and \( n_j \in N \) is the first node that exists in any thread originating from \( p \), then \( n_i ≺ n_j \).

Rule 2: If a node \( n_j \in N \) follows next after a ParallelEnd node \( p \) and \( n_k \in N \) is the last node in any thread joining with \( p \), then \( n_k ≺ n_j \).

Rule 3: If a node \( n_i \in N \) and other node \( n_j \in N \) are two consecutive concurrent nodes in a thread originating from a ParallelStart node \( p \) where \( n_i \) exists before \( n_j \) in the thread, then \( n_i ≺ n_j \).

Figure 2 shows a simplified model composed of concurrent and non-concurrent nodes. The nodes in the subpaths \([1, 2, 3]\) and \([8, 9, 10]\) are non-concurrent, and the nodes in the subpaths \([5, 7]\) and \([4, 6]\) are concurrent. There are six precedence rules in the graph \( G: 3 ≺ 5, 3 ≺ 4, 7 ≺ 8, 4 ≺ 6, 5 ≺ 7 \) and \( 6 ≺ 8 \). If we consider only the paths that comply with the six precedence relations and traverse all the nodes, we obtain \( 4!/2!*2! = 6 \) paths. This simple example shows how easy it is to face an explosion of candidate test paths. For example, if we add only one extra parallel subpath with two nodes, we would have \( 6!/2!*2!*2! = 90 \) paths. Moreover, the presence of loops and Condition nodes among concurrent nodes results in even more paths and hence, it may be impossible to consider all the possible test paths during acceptance testing due to limited resources.

AGAC avoids the generation of the entire set of possible test paths by selecting a concurrent subpath that maximizes the number of threads interleavings by exercising, in sequence, actions that belong to different threads (or SelPath). We select SelPath as the one in which the sequence of concurrent nodes corresponds to their breadth-first traversal since breadth-first traversal, by construction, selects subsequent activities belonging to different threads and ensures all precedence relationships among the nodes. The precedence relationships are satisfied because a breadth-first traversal of \( G \) starts at the ParallelStart node, and explores all of the immediately next nodes at the present depth prior to proceeding with the nodes at the next depth level. In addition, AGAC enables engineers to specify the maximum number of times a node belonging to a loop should be visited.
**III. GENERATED ACCEPTANCE CRITERIA IN GHERKIN**

<table>
<thead>
<tr>
<th>Feature: Perform a Settlement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background:</strong> Given SettlementPlatform.allInstances() -&gt; forAll (t : T, t.isInitialized is equal to true)</td>
</tr>
<tr>
<td># The intent “Create” was identified by analyzing the inputs and outputs</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Create</td>
</tr>
<tr>
<td>Given plx of type Participant Settlement Ins does not exist in P of type Participant</td>
</tr>
<tr>
<td>Then pInx exists in P</td>
</tr>
<tr>
<td># The intent “Send” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Send</td>
</tr>
<tr>
<td>Given plx of type Participant Settlement Ins does not exist in P of type Participant</td>
</tr>
<tr>
<td>Then pInx exists in P</td>
</tr>
<tr>
<td># The intent “Validate” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Validate</td>
</tr>
<tr>
<td>Given Inx of type T2S Settlement Ins does not exist in T2S of type Settlement Platform</td>
</tr>
<tr>
<td>Then T2S Receive and Generate Instruction</td>
</tr>
<tr>
<td>When T2S Receive and Generate Instruction</td>
</tr>
<tr>
<td>Then Inx exists in T2S</td>
</tr>
<tr>
<td>And the property State of Inx is equal to “ToValidate”</td>
</tr>
<tr>
<td># The intent “Receive” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Receive</td>
</tr>
<tr>
<td>Given Inx of type T2S Settlement Ins does not exist in T2S of type Settlement Platform</td>
</tr>
<tr>
<td>Then T2S Send Notification and Generate Instruction</td>
</tr>
<tr>
<td>When T2S Send Notification and Generate Instruction</td>
</tr>
<tr>
<td>Then Inx exists in T2S</td>
</tr>
<tr>
<td>And the property State of Inx is equal to “ToValidate”</td>
</tr>
<tr>
<td># The intent “Send” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Send</td>
</tr>
<tr>
<td>Given plx of type Participant Settlement Ins does not exist in P of type Participant</td>
</tr>
<tr>
<td>Then pInx exists in P</td>
</tr>
<tr>
<td># The intent “Validate” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Validate</td>
</tr>
<tr>
<td>Given Inx of type T2S Settlement Ins does not exist in T2S of type Settlement Platform</td>
</tr>
<tr>
<td>Then T2S Validate Inx</td>
</tr>
<tr>
<td>When T2S Validate Inx</td>
</tr>
<tr>
<td>Then T2S validated Inx</td>
</tr>
<tr>
<td># Passed by the Condition node “Inx.State is equal to ‘Valid’”</td>
</tr>
<tr>
<td># Passed by the Parallel Start node</td>
</tr>
<tr>
<td># The intent “Update” was identified by analyzing the inputs and outputs</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Update</td>
</tr>
<tr>
<td>Given plx of type Participant Settlement Ins exists in P of type Participant</td>
</tr>
<tr>
<td>Then pInx exists in T2S</td>
</tr>
<tr>
<td>And the property State of Inx is equal to “Valid”</td>
</tr>
<tr>
<td># Passed by the Merge node Merge1. Thread 1</td>
</tr>
<tr>
<td># Passed by the Merge node Merge2. Thread 2</td>
</tr>
<tr>
<td># The intent “Create” was identified by analyzing the inputs and outputs</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Create</td>
</tr>
<tr>
<td>Given Inx of type T2S Settlement Ins does not exist in T2S of type Settlement Platform</td>
</tr>
<tr>
<td>Then T2S Receive and Generate Instruction</td>
</tr>
<tr>
<td>When T2S Receive and Generate Instruction</td>
</tr>
<tr>
<td>Then Inx exists in T2S</td>
</tr>
<tr>
<td>And the property State of Inx is equal to “ToValidate”</td>
</tr>
<tr>
<td># The intent “Receive” was identified by analyzing the verb</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Receive</td>
</tr>
<tr>
<td>Given Inx of type T2S Settlement Ins does not exist in T2S of type Settlement Platform</td>
</tr>
<tr>
<td>Then T2S Send Notification</td>
</tr>
<tr>
<td>When T2S Send Notification</td>
</tr>
<tr>
<td>Given notif of type Participant Notification does not exists in P of type Participant</td>
</tr>
<tr>
<td>Then notif exists in T2S</td>
</tr>
<tr>
<td>And the property State of Inx is equal to “ToValidate”</td>
</tr>
<tr>
<td># Passed by the Exit node “FlowFinal”. Thread 1</td>
</tr>
<tr>
<td># Passed by the Exit node “FlowFinal”. Thread 2</td>
</tr>
<tr>
<td><strong>Scenario:</strong> Receive and Generate Instruction</td>
</tr>
<tr>
<td>Given plx of type Participant Settlement Ins does not exist in P of type Participant</td>
</tr>
<tr>
<td>Then the property State of Inx is equal to “Settled”</td>
</tr>
<tr>
<td># Passed by the Exit node “FlowFinal”. Thread 1</td>
</tr>
<tr>
<td># Passed by the Exit node “FlowFinal”. Thread 2</td>
</tr>
</tbody>
</table>

---

**Listing 2. Acceptance criterion related to path p₂**

**Listing 1. Acceptance criterion related to path p₁**

---

2
Feature: Perform a Settlement

Scenario: Send settlement Instruction
  Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
  When T2S Send settlement Instruction
  # The intent "Send" was identified by analyzing the inputs and outputs
  # Passed by the Condition node "Inx.State is equal to Valid"

Scenario: Receive and Generate Instruction
  Given pInx of type Participant Settlement Ins does not exists in P of type Participant
  Then pInx exists in P
  # The intent "Receive" was identified by analyzing the verb
  # Passed by the Merge node Merge2

Scenario: Validate Ins
  Given notif of type Participant Notification exists in T2S of type Settlement Platform
  When T2S Process Instruction Rejection
  Then the property State of Inx is equal to "Rejected"
  # The intent "Create" was identified by analyzing the inputs and outputs
  # Passed by the Exit node "FlowFinal"

Scenario: Receive and Generate Instruction
  Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
  When T2S Receive and Generate Instruction
  Then T2S validated Inx
  # The intent "Validate" was identified by analyzing the verb

Scenario: Create
  Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal to true)
  # The intent "Create" was identified by analyzing the inputs and outputs
  # Passed by the Exit node "FlowFinal"

Scenario: Send settlement Instruction
  Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
  When T2S Send settlement Instruction
  Then Inx exists in T2S
  # The intent "Send" was identified by analyzing the inputs and outputs
  # Passed by the Condition node "Inx.State is equal to "ToValidate"

Scenario: Receive notification
  Given notif of type Participant Notification does not exists in P of type Participant
  Then pInx exists in P
  # The intent "Receive" was identified by analyzing the verb
  # Passed by the Merge node Merge2

Scenario: Send settlement Instruction
  Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
  When T2S Send settlement Instruction
  Then notif exists in T2S
  # The intent "Create" was identified by analyzing the inputs and outputs

Scenario: Send Notification
  Given pInx of type Participant Settlement Ins exists in P of type Participant
  When P Send settlement Instruction
  Then P sent pInx
  # The intent "Create" was identified by analyzing the inputs and outputs
  # Passed by the Exit node "FlowFinal"

Scenario: Receive notification
  Given notif of type Participant Notification exists in T2S of type Settlement Platform
  When T2S Receive notification
  Then notif exists in T2S
  # The intent "Receive" was identified by analyzing the verb
  # Passed by the Condition node "in/xnotifications"

Scenario: Send Notification
  Given notif of type Participant Notification does not exists in P of type Participant
  When P Receive notification
  Then notif exists in P
  # Passed by the Exit node "FlowFinal"

Listing 3. Acceptance criterion related to path p3

Listing 4. Acceptance criterion related to path p4