

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

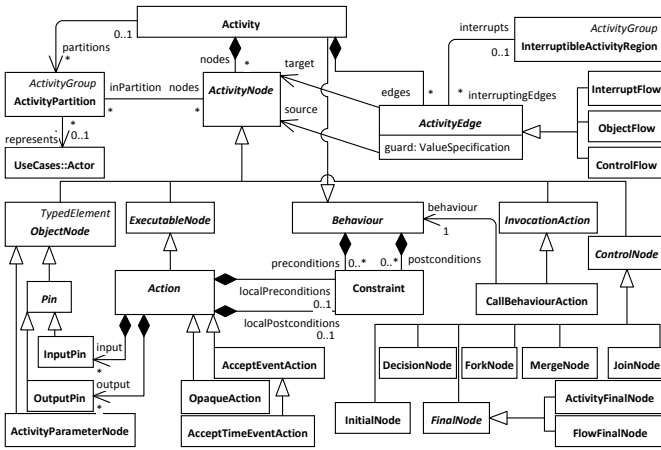


Fig. 1. Excerpt of the UML metamodel for Activity Diagrams

II. SELECTION OF CONCURRENT NODES

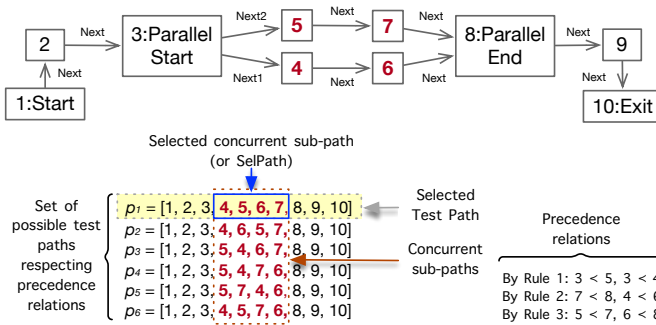


Fig. 2. Obtaining a test path containing a selected concurrent sub-path

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

```

1 Feature: Perform a Settlement
2 Background:
3 Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal ←
  to true)
4 # The intent "Create" was identified by analyzing the inputs and outputs
5 @Intent: Create
6 Scenario: Send settlement Instruction
7 Given pInx of type Participant Settlement Ins does not exists in P of type ←
  Participant
8 When P Send settlement Instruction
9 Then pInx exists in P
10 # The intent "Send" was identified by analyzing the verb
11 @Intent: Send
12 Scenario: Send settlement Instruction
13 Given pInx of type Participant Settlement Ins exists in P of type Participant
14 When P Send settlement Instruction
15 Then P sent pInx
16 # The intent "Create" was identified by analyzing the inputs and outputs
17 @Intent: Create
18 Scenario: Receive and Generate Instruction
19 Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement ←
  Platform
20 When T2S Receive and Generate Instruction
21 Then Inx exists in T2S
22 And the property State of Inx is equal to "ToValidate"
23 # The intent "Receive" was identified by analyzing the verb
24 @Intent: Receive
25 Scenario: Receive and Generate Instruction
26 Given pInx of type Participant Settlement Ins does not exists in T2S of type ←
  Settlement Platform
27 When T2S Receive and Generate Instruction
28 Then pInx exists in T2S
29 # The intent "Validate" was identified by analyzing the verb
30 @Intent: Validate
31 Scenario: Validate Ins
32 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
33 When T2S Validate Ins
34 Then T2S validated Inx
35 # Passed by the Condition node "Inx.State is equal to Valid"
36 # Passed by the Parallel Start node
37 # The intent "Update" was identified by analyzing the inputs and outputs
38 @Intent: Update
39 Scenario: Run Matching Process. Thread 1
40 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
41 And "Inx.State_is_equal_to_Valid" is True
42 And the property State of Inx is equal to "Valid"
43 When T2S Run Matching Process
44 Then the property State of Inx is equal to "Matched"
45 # Passed by the Merge node Merge2. Thread 2
46 # Passed by the Condition node "Inx.SettlementDate > T2S.CurrentDate". Thread 1
47 # The intent "Create" was identified by analyzing the inputs and outputs
48 @Intent: Create
49 Scenario: Send Notification. Thread 2
50 Given notif of type Participant Notification does not exists in T2S of type ←
  Settlement Platform
51 And "Inx.SettlementDate_>_T2S.CurrentDate" is equal to "Yes"
52 When T2S Send Notification
53 Then notif exists in T2S
54 # The intent "Send" was identified by analyzing the verb
55 @Intent: Send
56 Scenario: Send Notification. Thread 2
57 Given notif of type Participant Notification exists in T2S of type Settlement ←
  Platform
58 And "Inx.SettlementDate_>_T2S.CurrentDate" is equal to "Yes"
59 When T2S Send Notification
60 Then T2S sent notif
61 # Passed by the Event "Inx.SettlementDate starts". Thread 1
62 # The intent "Receive" was identified by analyzing the verb
63 @Intent: Receive
64 Scenario: Receive notification. Thread 2
65 Given notif of type Participant Notification does not exists in P of type ←
  Participant
66 And the event "Inx.SettlementDate_starts" happened
67 When P Receive notification
68 Then notif exists in P
69 # Passed by the Merge node Merge1. Thread 1
70 # Passed by the Exit node "FlowFinal". Thread 2
71 # The intent "Update" was identified by analyzing the inputs and outputs
72 @Intent: Update
73 Scenario: Settle Instruction. Thread 1
74 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
75 And the property State of Inx is equal to "Matched"
76 When T2S Settle Instruction
77 Then the property State of Inx is equal to "Settled"
78 # Passed by the Merge node Merge2. Thread 1
79 # The intent "Create" was identified by analyzing the inputs and outputs
80 @Intent: Create
81 Scenario: Send Notification. Thread 1
82 Given notif of type Participant Notification does not exists in T2S of type ←
  Settlement Platform
83 When T2S Send Notification
84 Then notif exists in T2S
85 # The intent "Send" was identified by analyzing the verb
86 @Intent: Send
87 Scenario: Send Notification. Thread 1
88 Given notif of type Participant Notification exists in T2S of type Settlement ←
  Platform
89 When T2S Send Notification
90 Then T2S sent notif
91 # The intent "Receive" was identified by analyzing the verb
92 @Intent: Receive
93 Scenario: Receive notification. Thread 1
94 Given notif of type Participant Notification does not exists in P of type ←
  Participant
95 When P Receive notification
96 Then notif exists in P
97 # Passed by the Exit node "FlowFinal". Thread 1

```

Listing 1. Acceptance criterion related to path p_1

```

1 Feature: Perform a Settlement
2 Background:
3 Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal ←
  to true)
4 # The intent "Create" was identified by analyzing the inputs and outputs
5 @Intent: Create
6 Scenario: Send settlement Instruction
7 Given pInx of type Participant Settlement Ins does not exists in P of type ←
  Participant
8 When P Send settlement Instruction
9 Then pInx exists in P
10 # The intent "Send" was identified by analyzing the verb
11 @Intent: Send
12 Scenario: Send settlement Instruction
13 Given pInx of type Participant Settlement Ins exists in P of type Participant
14 When P Send settlement Instruction
15 Then P sent pInx
16 # The intent "Create" was identified by analyzing the inputs and outputs
17 @Intent: Create
18 Scenario: Receive and Generate Instruction
19 Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement ←
  Platform
20 When T2S Receive and Generate Instruction
21 Then Inx exists in T2S
22 And the property State of Inx is equal to "ToValidate"
23 # The intent "Receive" was identified by analyzing the verb
24 @Intent: Receive
25 Scenario: Receive and Generate Instruction
26 Given pInx of type Participant Settlement Ins does not exists in T2S of type ←
  Settlement Platform
27 When T2S Receive and Generate Instruction
28 Then pInx exists in T2S
29 # The intent "Validate" was identified by analyzing the verb
30 @Intent: Validate
31 Scenario: Validate Ins
32 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
33 When T2S Validate Ins
34 Then T2S validated Inx
35 # Passed by the Condition node "Inx.State is equal to Valid"
36 # Passed by the Parallel Start node
37 # The intent "Update" was identified by analyzing the inputs and outputs
38 @Intent: Update
39 Scenario: Run Matching Process. Thread 1
40 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
41 And "Inx.State_is_equal_to_Valid" is True
42 And the property State of Inx is equal to "Valid"
43 When T2S Run Matching Process
44 Then the property State of Inx is equal to "Matched"
45 # Passed by the Merge node Merge2. Thread 2
46 # Passed by the Condition node "Inx.SettlementDate > T2S.CurrentDate". Thread 1
47 # The intent "Create" was identified by analyzing the inputs and outputs
48 @Intent: Create
49 Scenario: Send Notification. Thread 2
50 Given notif of type Participant Notification does not exists in T2S of type ←
  Settlement Platform
51 And "Inx.SettlementDate_>_T2S.CurrentDate" is equal to "No"
52 When T2S Send Notification
53 Then notif exists in T2S
54 # The intent "Send" was identified by analyzing the verb
55 @Intent: Send
56 Scenario: Send Notification. Thread 2
57 Given notif of type Participant Notification exists in T2S of type Settlement ←
  Platform
58 And "Inx.SettlementDate_>_T2S.CurrentDate" is equal to "No"
59 When T2S Send Notification
60 Then T2S sent notif
61 # Passed by the Merge node Merge1. Thread 1
62 # The intent "Receive" was identified by analyzing the verb
63 @Intent: Receive
64 Scenario: Receive notification. Thread 2
65 Given notif of type Participant Notification does not exists in P of type ←
  Participant
66 When P Receive notification
67 Then notif exists in P
68 # The intent "Update" was identified by analyzing the inputs and outputs
69 @Intent: Update
70 Scenario: Settle Instruction. Thread 1
71 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
72 And the property State of Inx is equal to "Matched"
73 When T2S Settle Instruction
74 Then the property State of Inx is equal to "Settled"
75 # Passed by the Exit node "FlowFinal". Thread 2
76 # Passed by the Merge node Merge2. Thread 1
77 # The intent "Create" was identified by analyzing the inputs and outputs
78 @Intent: Create
79 Scenario: Send Notification. Thread 1
80 Given notif of type Participant Notification does not exists in T2S of type ←
  Settlement Platform
81 When T2S Send Notification
82 Then notif exists in T2S
83 # The intent "Send" was identified by analyzing the verb
84 @Intent: Send
85 Scenario: Send Notification. Thread 1
86 Given notif of type Participant Notification exists in T2S of type Settlement ←
  Platform
87 When T2S Send Notification
88 Then T2S sent notif
89 # The intent "Receive" was identified by analyzing the verb
90 @Intent: Receive
91 Scenario: Receive notification. Thread 1
92 Given notif of type Participant Notification does not exists in P of type ←
  Participant
93 When P Receive notification
94 Then notif exists in P
95 # Passed by the Exit node "FlowFinal". Thread 1

```

Listing 2. Acceptance criterion related to path p_2

```

1 Feature: Perform a Settlement
2 Background:
3 Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal ←
to true)
4 # The intent "Create" was identified by analyzing the inputs and outputs
5 @Intent: Create
6 Scenario: Send settlement Instruction
7 Given pInx of type Participant Settlement Ins does not exists in P of type ←
Participant
8 When P Send settlement Instruction
9 Then pInx exists in P
10 # The intent "Send" was identified by analyzing the verb
11 @Intent: Send
12 Scenario: Send settlement Instruction
13 Given pInx of type T2S Settlement Ins exists in P of type Participant
14 When P Send settlement Instruction
15 Then P sent pInx
16 # The intent "Create" was identified by analyzing the inputs and outputs
17 @Intent: Create
18 Scenario: Receive and Generate Instruction
19 Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement ←
Platform
20 When T2S Receive and Generate Instruction
21 Then Inx exists in T2S
22 And the property State of Inx is equal to "ToValidate"
23 # The intent "Receive" was identified by analyzing the verb
24 @Intent: Receive
25 Scenario: Receive and Generate Instruction
26 Given pInx of type Participant Settlement Ins does not exists in T2S of type ←
Settlement Platform
27 When T2S Receive and Generate Instruction
28 Then pInx exists in T2S
29 # The intent "Validate" was identified by analyzing the verb
30 @Intent: Validate
31 Scenario: Validate Ins
32 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
33 When T2S Validate Ins
34 Then T2S validated Inx
35 # Passed by the Condition node "Inx.State is equal to Valid"
36 # The intent "Update" was identified by analyzing the inputs and outputs
37 @Intent: Update
38 Scenario: Process Instruction Rejection
39 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
40 And "Inx.State_is_equal_to_Valid" is False
41 When T2S Process Instruction Rejection
42 Then the property State of Inx is equal to "Rejected"
43 # Passed by the Merge node Merge2
44 # The intent "Create" was identified by analyzing the inputs and outputs
45 @Intent: Create
46 Scenario: Send Notification
47 Given notif of type Participant Notification does not exists in T2S of type ←
Settlement Platform
48 When T2S Send Notification
49 Then notif exists in T2S
50 # The intent "Send" was identified by analyzing the verb
51 @Intent: Send
52 Scenario: Send Notification
53 Given notif of type Participant Notification exists in T2S of type Settlement ←

```

```

Platform
54 When T2S Send Notification
55 Then T2S sent notif
56 # The intent "Receive" was identified by analyzing the verb
57 @Intent: Receive
58 Scenario: Receive notification
59 Given notif of type Participant Notification does not exists in P of type ←
Participant
60 When P Receive notification
61 Then notif exists in P
62 # Passed by the Exit node "FlowFinal"

```

Listing 3. Acceptance criterion related to path p_3

```

1 Feature: Perform a Settlement
2 # The intent "Interrupt" was identified by analyzing the region and type of the ←
outgoing flow of the event
3 @Intent: Interrupt
4 Scenario: X days passed
5 Given Run Matching Process is running in T2S of type Settlement Platform
6 When the event "X_days_passed" happens in T2S
7 Then Run Matching Process is interrupted in T2S
8 # The intent "Update" was identified by analyzing the inputs and outputs
9 @Intent: Update
10 Scenario: Process Instruction Rejection
11 Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
12 When T2S Process Instruction Rejection
13 Then the property State of Inx is equal to "Rejected"
14 # Passed by the Merge node Merge2
15 # The intent "Create" was identified by analyzing the inputs and outputs
16 @Intent: Create
17 Scenario: Send Notification
18 Given notif of type Participant Notification does not exists in T2S of type ←
Settlement Platform
19 When T2S Send Notification
20 Then notif exists in T2S
21 # The intent "Send" was identified by analyzing the verb
22 @Intent: Send
23 Scenario: Send Notification
24 Given notif of type Participant Notification exists in T2S of type Settlement ←
Platform
25 When T2S Send Notification
26 Then T2S sent notif
27 # The intent "Receive" was identified by analyzing the verb
28 @Intent: Receive
29 Scenario: Receive notification
30 Given notif of type Participant Notification does not exists in P of type ←
Participant
31 When P Receive notification
32 Then notif exists in P
33 # Passed by the Exit node "FlowFinal"

```

Listing 4. Acceptance criterion related to path p_4