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

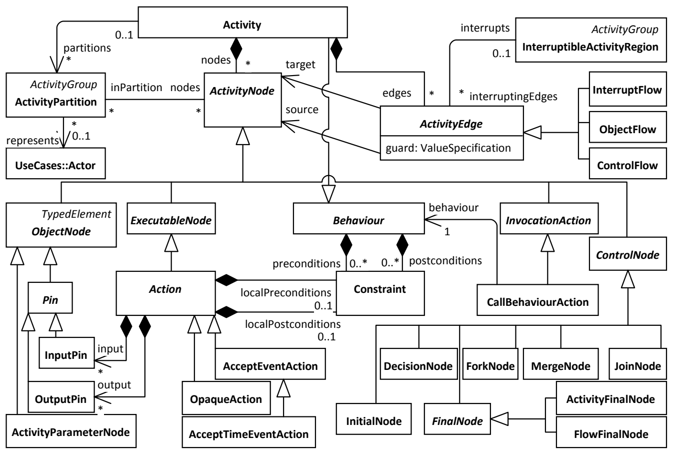

Fig. 1. Excerpt of the UML metamodel for Activity Diagrams

II. Selection of concurrent nodes

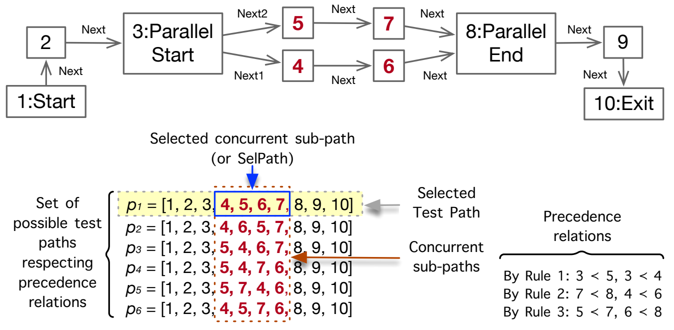
2
Next
3:Parallel Start
Next2
5
Next
7
Next
8:Parallel End
Next
9
Next
1:Start
Next1
4
Next
6
Next
Next
10:Exit
Selected concurrent sub-path (or SelPath)
Set of possible test paths respecting precedence relations
p1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
p2 = [1, 2, 3, 4, 6, 5, 7, 8, 9, 10]
p3 = [1, 2, 3, 5, 4, 6, 7, 8, 9, 10]
p4 = [1, 2, 3, 5, 4, 7, 6, 8, 9, 10]
p5 = [1, 2, 3, 5, 7, 4, 6, 8, 9, 10]
p6 = [1, 2, 3, 4, 5, 7, 6, 8, 9, 10]
Selected Test Path
Concurrent sub-paths
Precedence relations
By Rule 1: 3 < 5, 3 < 4
By Rule 2: 7 < 8, 4 < 6
By Rule 3: 5 < 7, 6 < 8

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 "$\prec$", 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 $n_i \prec n_j$.

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 $n_k \prec n_j$.

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 $n_i \prec n_j$.

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: $3 \prec 5$, $3 \prec 4$, $7 \prec 8$, $4 \prec 6$, $5 \prec 7$ and $6 \prec 8$. 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

```
Feature: Perform a Settlement
Background:
Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal to true)
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send settlement Instruction
Given pInx of type Participant Settlement Ins does not exists in P of type Participant
When P Send settlement Instruction
Then pInx exists in P
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send settlement Instruction
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
@Intent: Create
Scenario: Receive and Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Receive and Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive and Generate Instruction
Given pInx of type Participant Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Receive and Generate Instruction
Then pInx exists in T2S
# The intent "Validate" was identified by analyzing the verb
@Intent: Validate
Scenario: Validate Ins
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
When T2S Validate Ins
Then T2S validated Inx
# Passed by the Condition node "Inx.State is equal to Valid"
# Passed by the Parallel Start node
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Run Matching Process. Thread 1
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
And "Inx.State␣is␣equal␣to␣Valid" is True
And the property State of Inx is equal to "Valid"
When T2S Run Matching Process
Then the property State of Inx is equal to "Matched"
# Passed by the Merge node Merge2. Thread 2
# Passed by the Condition node "Inx.SettlementDate > T2S.CurrentDate". Thread 1
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification. Thread 2
Given notif of type Participant Notification does not exists in T2S of type Settlement Platform
And "Inx.SettlementDate␣>␣T2S.CurrentDate" is equal to "Yes"
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification. Thread 2
Given notif of type Participant Notification exists in T2S of type Settlement Platform
And "Inx.SettlementDate␣>␣T2S.CurrentDate" is equal to "Yes"
When T2S Send Notification
Then T2S sent notif
# Passed by the Event "Inx.SettlementDate starts". Thread 1
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive notification. Thread 2
Given notif of type Participant Notification does not exists in P of type Participant
And the event "Inx.SettlementDate␣starts" happened
When P Receive notification
Then notif exists in P
# Passed by the Merge node Merge1. Thread 1
# Passed by the Exit node "FlowFinal". Thread 2
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Settle Instruction. Thread 1
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
And the property State of Inx is equal to "Matched"
When T2S Settle Instruction
Then the property State of Inx is equal to "Settled"
# Passed by the Merge node Merge2. Thread 1
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification. Thread 1
Given notif of type Participant Notification does not exists in T2S of type Settlement Platform
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification. Thread 1
Given notif of type Participant Notification exists in T2S of type Settlement Platform
When T2S Send Notification
Then T2S sent notif
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive notification. Thread 1
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". Thread 1
```

Listing 1. Acceptance criterion related to path p_1

```
Feature: Perform a Settlement
Background:
Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal to true)
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send settlement Instruction
Given pInx of type Participant Settlement Ins does not exists in P of type Participant
When P Send settlement Instruction
Then pInx exists in P
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send settlement Instruction
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
@Intent: Create
Scenario: Receive and Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Receive and Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive and Generate Instruction
Given pInx of type Participant Settlement Ins does not exists in T2S of type Settlement Platform
When T2S Receive and Generate Instruction
Then pInx exists in T2S
# The intent "Validate" was identified by analyzing the verb
@Intent: Validate
Scenario: Validate Ins
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
When T2S Validate Ins
Then T2S validated Inx
# Passed by the Condition node "Inx.State is equal to Valid"
# Passed by the Parallel Start node
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Run Matching Process. Thread 1
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
And "Inx.State␣is␣equal␣to␣Valid" is True
And the property State of Inx is equal to "Valid"
When T2S Run Matching Process
Then the property State of Inx is equal to "Matched"
# Passed by the Merge node Merge2. Thread 2
# Passed by the Condition node "Inx.SettlementDate > T2S.CurrentDate". Thread 1
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification. Thread 2
Given notif of type Participant Notification does not exists in T2S of type Settlement Platform
And "Inx.SettlementDate␣>␣T2S.CurrentDate" is equal to "No"
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification. Thread 2
Given notif of type Participant Notification exists in T2S of type Settlement Platform
And "Inx.SettlementDate␣>␣T2S.CurrentDate" is equal to "No"
When T2S Send Notification
Then T2S sent notif
# Passed by the Merge node Merge1. Thread 1
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive notification. Thread 2
Given notif of type Participant Notification does not exists in P of type Participant
When P Receive notification
Then notif exists in P
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Settle Instruction. Thread 1
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
And the property State of Inx is equal to "Matched"
When T2S Settle Instruction
Then the property State of Inx is equal to "Settled"
# Passed by the Exit node "FlowFinal". Thread 2
# Passed by the Merge node Merge2. Thread 1
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification. Thread 1
Given notif of type Participant Notification does not exists in T2S of type Settlement Platform
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification. Thread 1
Given notif of type Participant Notification exists in T2S of type Settlement Platform
When T2S Send Notification
Then T2S sent notif
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive notification. Thread 1
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". Thread 1
```

Listing 2. Acceptance criterion related to path p_2

```
Feature: Perform a Settlement
Background:
Given SettlementPlatform.allInstances() -> forAll (t / t.isInitialised is equal ↩
       to true)
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send settlement Instruction
Given pInx of type Participant Settlement Ins does not exists in P of type ↩
       Participant
When P Send settlement Instruction
Then pInx exists in P
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send settlement Instruction
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
@Intent: Create
Scenario: Receive and Generate Instruction
Given Inx of type T2S Settlement Ins does not exists in T2S of type Settlement ↩
       Platform
When T2S Receive and Generate Instruction
Then Inx exists in T2S
And the property State of Inx is equal to "ToValidate"
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive and Generate Instruction
Given pInx of type Participant Settlement Ins does not exists in T2S of type ↩
       Settlement Platform
When T2S Receive and Generate Instruction
Then pInx exists in T2S
# The intent "Validate" was identified by analyzing the verb
@Intent: Validate
Scenario: Validate Ins
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
When T2S Validate Ins
Then T2S validated Inx
# Passed by the Condition node "Inx.State is equal to Valid"
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Process Instruction Rejection
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
And "Inx.State␣is␣equal␣to␣Valid" is False
When T2S Process Instruction Rejection
Then the property State of Inx is equal to "Rejected"
# Passed by the Merge node Merge2
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification
Given notif of type Participant Notification does not exists in T2S of type ↩
       Settlement Platform
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification
Given notif of type Participant Notification exists in T2S of type Settlement ↩
       Platform
When T2S Send Notification
Then T2S sent notif
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive 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 p_3

```
Feature: Perform a Settlement
# The intent "Interrupt" was identified by analyzing the region and type of the ↩
       outgoing flow of the event
@Intent: Interrupt
Scenario: X days passed
Given Run Matching Process is running in T2S of type Settlement Platform
When the event "X␣days␣passed" happens in T2S
Then Run Matching Process is interrupted in T2S
# The intent "Update" was identified by analyzing the inputs and outputs
@Intent: Update
Scenario: Process Instruction Rejection
Given Inx of type T2S Settlement Ins exists in T2S of type Settlement Platform
When T2S Process Instruction Rejection
Then the property State of Inx is equal to "Rejected"
# Passed by the Merge node Merge2
# The intent "Create" was identified by analyzing the inputs and outputs
@Intent: Create
Scenario: Send Notification
Given notif of type Participant Notification does not exists in T2S of type ↩
       Settlement Platform
When T2S Send Notification
Then notif exists in T2S
# The intent "Send" was identified by analyzing the verb
@Intent: Send
Scenario: Send Notification
Given notif of type Participant Notification exists in T2S of type Settlement ↩
       Platform
When T2S Send Notification
Then T2S sent notif
# The intent "Receive" was identified by analyzing the verb
@Intent: Receive
Scenario: Receive 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 4. Acceptance criterion related to path p_4