Report of the PSSM 1.0 FTF 2 Finalization Task Force to the OMG Platform Technical Committee

13 November 2018

Document Number:	**ptc/18-11-01**
Task Force Chair(s):	**Ed Seidewitz, Model Driven Solutions (ed-s@modeldriven.com)** **Jeremie Tatibouet, Commissariat a l Energie Atomique-CEA (jeremie.tatibouet@cea.fr)**
Chartered:	**8 December 2017 — Burlingame, CA , USA**
Comments Due:	**5 March 2018**
Expiry Due:	**21 December 2018**
JIRA Project Prefix:	**PSSM_**
Document Template:	omg/2013-05-01

Table of Contents

- **Deliverables**
  - *Publication Directions*
  - *Specification*
  - *Supporting Documents*
  - *Machine-consumable documents*
- **IPR Mode**
- **FTF Membership**
- **Disposition Summary**
- **Voting Record**
- **Summary of Changes Made**
- **Disposition: Resolved**
  - *OMG Issue PSSM_-2*
    Title: Synchronous operation call on an active object ends if the corresponding call event occurrence was deferred
  - *OMG Issue PSSM_-4*
    Title: PSSM shall be aligned with fUML 1.3 and PSCS 1.1
  - *OMG Issue PSSM_-7*
    Title: In Testcase "Entering 011" T3 is not a completion transition
  - *OMG Issue PSSM_-8*
    Title: "Join 003" test case state machine diagram appears to be invalid
  - *OMG Issue PSSM_-9*
    Title: Example state machine appears to violate UML constraints
  - *OMG Issue PSSM_-10*
    Title: Typos in "Note" section of "9.3.3.12 Transition 019"
  - *OMG Issue PSSM_-12*
    Title: PSSM should align with fUML 1.4 and PSCS 1.2
  - *OMG Issue PSSM_-23*
    Title: Incorrect transition numbering in Deferred007_Test
- **Disposition: Deferred**
  - *OMG Issue PSSM_-1*
    Title: Tests that send multiple signals are not correct
  - *OMG Issue PSSM_-3*
    Title: PSSM implementation shall conform to bUML
  - *OMG Issue PSSM_-6*
    Title: Notation for entry, do and exit behaviors is wrong
- **Disposition: Transfered**
- **Disposition: Closed; No Change**
- **Disposition: Closed; Out Of Scope**
- **Disposition: Duplicate or Merged**
  - *OMG Issue PSSM_-11*
    Title: "Join 003" has (invalid) triggers on transitions entering join

Deliverables

Publication Directions

This section contains information about how OMG staff should publish the resulting specification on the OMG formal specification web page. The chosen acronym and version number determine the URL to be used; for instance, if the acronym chosen is "ABCD" and the version number is "1.2", the normative specification documents will be available under the URL https://www.omg.org/spec/ABCD/1.2/.

Documents in this section are classified as one of:

Normative:	A document that specifies how a conforming implementation shall behave (but may contain specifically-labelled non-normative sections).
Informative:	Non-normative material aimed at users of the specification, such as examples or non-normative guidelines.
Ancillary:	Material intended to help OMG Members reviewing the report or creating future revisions of the specification, such as a change-tracked specification showing issue resolutions applied, editable source of models stored in proprietary formats, or programs used to build parts of the specification from machine-readable sources. The specification shall be usable without reference to the ancillary files, which should not be given URLs.

Specification

Chosen acronym: PSSM
Version number: 1.0

Title:	**Revised specification (clean)**
Doc Number:	ptc/18-11-02
Status:	Normative
URL:	https://www.omg.org/spec/PSSM/1.0

Title:	**Precise Semantics of State Machines (PSSM) v1.0 Beta 2 Specification (change bars)**
Doc Number:	ptc/18-11-03
Status:	Informative

Supporting Documents

Title: **PSSM 1.0 Ancillary Archive (Zip File)**
Doc Number: ptc/18-11-07
Status: Ancillary

Title: **Issue PSSM_-2 Attachment Deferred007_Test.PNG**
Doc Number: ptc/18-11-07
Filename: PSSM_-2/Deferred007_Test.PNG (zip entry)
Status: Ancillary

Title: **Issue PSSM_-2 Attachment Figure-9.101.png**
Doc Number: ptc/18-11-07
Filename: PSSM_-2/Figure-9.101.png (zip entry)
Status: Ancillary

Title: **Issue PSSM_-2 Attachment Figure-9.102.png**
Doc Number: ptc/18-11-07
Filename: PSSM_-2/Figure-9.102.png (zip entry)
Status: Ancillary

Title: **Issue PSSM_-2 Attachment New-Tester-Behavior.png**
Doc Number: ptc/18-11-07
Filename: PSSM_-2/New-Tester-Behavior.png (zip entry)
Status: Ancillary

Title: **Issue PSSM_-2 Attachment Old-Tester-Behavior.png**
Doc Number: ptc/18-11-07
Filename: PSSM_-2/Old-Tester-Behavior.png (zip entry)
Status: Ancillary

Title: **Issue PSSM_-8 Attachment Join003_Test-v1.1.svg**
Doc Number: ptc/18-11-07
Filename: PSSM_-8/Join003_Test-v1.1.svg (zip entry)
Status: Ancillary

Title: **Issue PSSM_-9 Attachment Transition023_Test-v1.0.svg**
Doc Number: ptc/18-11-07
Filename: PSSM_-9/Transition023_Test-v1.0.svg (zip entry)
Status: Ancillary

Title: **Issue PSSM_-23 Attachment Deferred007_Test.svg**
Doc Number: ptc/18-11-07
Filename: PSSM_-23/Deferred007_Test.svg (zip entry)
Status: Ancillary

Machine-consumable documents

This section contains one entry for each machine-consumable file accompanying the submission.

Authors should assign a version stamp to each new normative or informative machine-consumable file that they create. Version stamps comprise 8 digits, and each newly-assigned version stamp for a particular specification must be greater than any previous version stamp for the same specification. One convenient way to achieve this is to concatenate the current year & two-digit month number with a two-digit sequence number within the month. Hence the sixth version created during May 2013 would have version stamp 20130506. Multiple files with different names should share the same version stamp where appropriate.

If many files are supplied, it may be convenient to package them in a single ZIP archive with a single OMG document number; however, each separate file should nevertheless still be given its own entry in this section.

For each machine-readable file, list any other files on which it depends (e.g. by inclusion), and whether it's normative, informative or ancillary (see above).

Each normative or informative machine-readable file is given a URL that allows it to be unambiguously referenced on the OMG formal specification web site. Ancillary files do not have URLs - they are not usually placed on the formal specification page.

Description:	**PSSM 1.0 Syntax Model XMI**
Doc Number:	ptc/18-11-04
Status:	Normative
URL:	https://www.omg.org/spec/PSSM/20181101/PSSM_Syntax.xmi

Description:	**PSSM 1.0 Semantics Model XMI**
Doc Number:	ptc/18-11-05
Status:	Normative
URL:	https://www.omg.org/spec/PSSM/20181101/PSSM_Semantics.xmi

Description:	**PSSM 1.0 Test Suite XMI**
Doc Number:	ptc/18-11-06
Status:	Normative
URL:	https://www.omg.org/spec/PSSM/20181101/PSSM_TestSuite.xmi

IPR Mode

Under OMG's IPR policy, every specification published after April 2013 must include a declaration of the terms under which anyone who contributed text to the specification must agree to license any patents that they control, and which they claim are essential to implementing the specification. This is known as the "IPR Mode" of the specification. It must be chosen from the list specified in the IPR policy, and once OMG's Board has selected the IPR Mode for a specification, it cannot subsequently be changed for any later RTF or FTF for that specification.

For RTFs or FTFs chartered before April 2013, a Legacy IPR mode must be selected for the specification being revised/finalised using the procedure in ipr/12-11-01 section 3.3.3, and listed below.

For RTFs or FTFs chartered after April 2013 for a specification without an IPR mode, the charter includes a Legacy IPR mode selected by the same procedure - that mode is listed bellow.

For a full, definitive statement of OMG's IPR policy, see: http://doc.omg.org/ipr.

IPR mode of base specification: Non-Assert

FTF Membership

Member	Organization	Role	Status
Manfred Koethe	88solutions	Voting Member	Charter
Jeff Smith	Multi Agency Collaboration Environment	Voting Member	Charter
Ed Seidewitz	Model Driven Solutions	Chair	Charter
Julio Medina	Universidad de Cantabria	Voting Member	Charter
Nerijus Jankevicius	No Magic, Inc.	Voting Member	Charter
Laura Hart	MITRE	Voting Member	Charter
Yves Bernard	Airbus Group	Voting Member	Charter
Robert Karban	NASA	Voting Member	Charter
Daniel Siegl	LieberLieber Software	Voting Member	Charter
Jeremie Tatibouet	Commissariat a l Energie Atomique-CEA	Chair	Charter
Axel Scheithauer	oose Innovative Informatik eG	Voting Member	Charter
Bran Selic	Simula Research Laboratory	Voting Member	Charter
John Butler	Auxilium Technology Group	Voting Member	Charter
Sam Mancarella	ModelFoundry	Voting Member	Charter

Disposition Summary

Disposition	Number Of Issues	Meaning Of Disposition
Resolved	8	The RTF/FTF agreed that there is a problem that needs fixing, and has proposed a resolution (which may or may not agree with any resolution the issue submitter proposed)
Deferred	3	The RTF/FTF agrees that there is a problem that needs fixing, but did not agree on a resolution and deferred its resolution to a future RTF/FTF.
Transfered	0	The RTF/FTF agrees to move the issue to another RTF/FTF.
Closed; No Change	0	The RTF/FTF decided that the issue report does not, in fact, identify a problem with this (or any other) OMG specification.
Closed; Out Of Scope	0	The RTF/FTF decided that the issue report is an enhancement request, and therefore out of scope for this or any future FTF or RTF working on this major version of the specification. The RTF/FTF has closed the issue without making any specification changes, but RFP or RFC submission teams may like to consider these enhancement requests when proposing future new major versions of the specification.
Duplicate or Merged	1	This issue is either an exact duplicate of another issue, or very closely related to another issue: see that issue for disposition.

Voting Record

Note: Issue numbers shown in the tables below should be prefixed with the JIRA Project Prefix (PSSM_-) to form fully qualified issue numbers.

Ballot No.	Closing Date	Number of Issues	Issues Included
2	8 November 2018	9	1, 2, 3, 4, 6, 7, 10, 11, 12
3	11 November 2018	3	8, 9, 23

The following tables detail the vote for each member in each ballot. The vote is described as follows:

Y Yes - Positive Vote

N No - Negative Vote

A Abstain

- Member Did Not Vote

Voting Record for Ballot No. 2

Voter	1	2	3	4	6	7	10	11	12
Manfred Koethe	Y	Y	Y	Y	Y	Y	Y	Y	Y
Yves Bernard	Y	Y	Y	Y	Y	Y	Y	Y	Y
John Butler	-	-	-	-	-	-	-	-	-
Jeremie Tatibouet	Y	Y	Y	Y	Y	Y	Y	Y	Y
Daniel Siegl	-	-	-	-	-	-	-	-	-
Laura Hart	-	-	-	-	-	-	-	-	-
Sam Mancarella	-	-	-	-	-	-	-	-	-
Jeff Smith	-	-	-	-	-	-	-	-	-
Robert Karban	Y	Y	Y	Y	Y	Y	Y	Y	Y
Nerijus Jankevicius	-	-	-	-	-	-	-	-	-
Bran Selic	Y	Y	Y	Y	Y	Y	Y	Y	Y
Julio Medina	Y	Y	Y	Y	Y	Y	Y	Y	Y
Ed Seidewitz	Y	Y	Y	Y	Y	Y	Y	Y	Y
Axel Scheithauer	Y	Y	Y	Y	Y	Y	Y	Y	Y

Voting Record for Ballot No. 3

Voter	8	9	23
Manfred Koethe	Y	Y	Y
Yves Bernard	Y	Y	Y
John Butler	-	-	-
Jeremie Tatibouet	Y	Y	Y
Daniel Siegl	-	-	-
Laura Hart	-	-	-
Sam Mancarella	-	-	-
Jeff Smith	-	-	-
Robert Karban	Y	Y	Y
Nerijus Jankevicius	-	-	-
Bran Selic	Y	Y	Y
Julio Medina	Y	Y	Y
Ed Seidewitz	Y	Y	Y
Axel Scheithauer	Y	Y	Y

Summary of Changes Made

The FTF made changes that:

- Corrected errors in the specification
- Increased the clarity of the specification
- Corrected features in order to improve implementability and adoption of the standard

Here is the FTF's categorization of the resolutions applied to the specification according to their impact on the clarity and precision of the specification:

Note: Issue numbers shown in the tables below should be prefixed with the JIRA Project Prefix (PSSM_-) to form fully qualified issue numbers.

Extent of Change	Number of Issues	Issues Included
Critical/Urgent — Fixed problems with normative parts of the specification which prevented implementation work.	0	
Significant — Fixed problems with normative parts of the specification that raised concern about implementability.	6	2, 4, 8, 9, 12, 23
Minor — Fixed minor problems with normative parts of the specification.	2	7, 10
Support Text — Changes to descriptive, explanatory, or supporting material.	0	

Disposition: Resolved

OMG Issue: PSSM_-2

Title:

Synchronous operation call on an active object ends if the corresponding call event occurrence was deferred

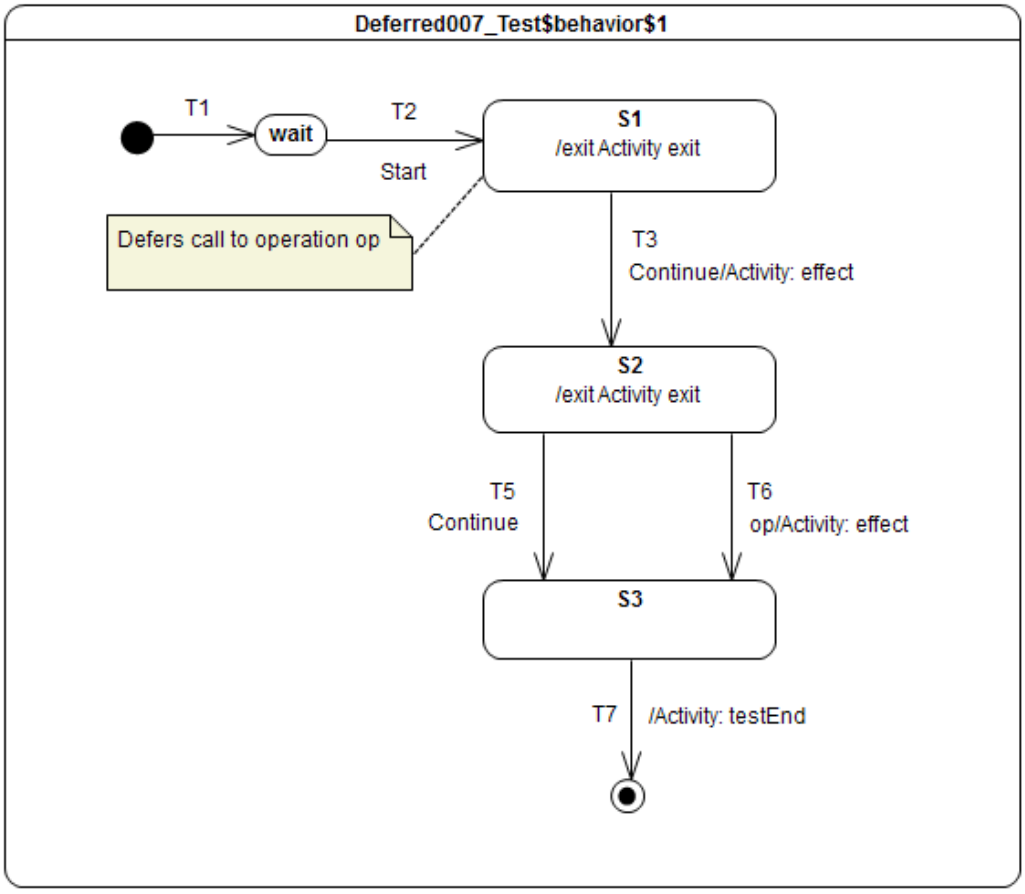
Summary:

FUML 1.3 provides a support for the CallEvent semantics. When an object performs a synchronous operation call on an active object then a call event occurrence is placed in the event pool of the target (i.e. the active object that is the target of the call). While the target has not accepted the call and the RTC initiated by the acceptance is not completed, the execution thread in which the caller executes remains suspended.

In PSSM context, the target of the call can be an active object whose executing a classifier behavior described thanks to a state machine. This state machine can have states that declare a call event as being deferred. This means that when the state is active and such call event is dispatched then it is going to be accepted by the state machine. The acceptance leads the call event occurrence to be placed in the deferred event pool of the active object.

The problem here is that since the call event was deferred, this implies the operation call was not performed by the target. Hence the caller shall not be released before the call event gets "undeferred" (i.e., return back to the regular event pool). However, this is not what is specified in the StateMachineEventAccepter (see section 8.5.2 in [PSSM 1.0b]) semantics. Indeed, the acceptance of the call event occurrence systematically leads to release the caller. Instead, the caller shall only get released if the call event occurrence is used to trigger one to many transitions.

The problem can be highlighted through the test case Deferred007_Test provided through the PSSM test suite.



In this test, the call event occurrence corresponding to the op operation is dispatched when in configuration S1. This implies the state machine accepts and defers the event occurrence. At the end of the RTC step, the tester (i.e., the object that emitted the call) shall remain suspended. This shall be maintained until the end of the RTC in which the transition T6 is fired. To demonstrate this semantics, Deferred007 Test shall be refactored.

Resolution Summary:

Update the StateMachineEventAccepter and test Deferred 007.

Rules that constrain the acceptance of an event by a state machine are specified in the accept operation of class StateMachineSemanticVisitor. The specification of this operation needs to be updated to ensure that the caller of an operation gets released only when the corresponding call event occurrence is accepted, not when it is deferred.

In addition, the test Deferred007_Test must be updated because it currently does not enable the assessment of the combined usage of a call event and deferral semantics. Indeed, as soon as S1 is deferred, the tester (i.e., the entity providing the stimulus) is suspended until the call event occurrence for the operation op is un-deferred and accepted. As Deferred007_Test will never receive a Continue signal event occurrence (since the tester is suspended), T3 will never have a chance to be fired, hence the state machine (and so the test) will remain stuck forever.

The issue can be resolved by giving S1 a do-activity behavior that is responsible for sending the Continue signal event occurrence to the test (i.e., the instance of Deferred007_Test). The Continue signal event occurrence will then be accepted by the state machine, T3 will be triggered and the call event occurrence will be un-deferred. This un-deferred event occurrence will then be used to trigger T6. Note, however, that this solution will only work if the call event occurrence is received before the Continue event occurrence.

Revised Text:

Specification Document

Update the specification document as follows.

8.5.2 State Machine Execution

Under "StateMachineEventAccepter", at the end of the second numbered item after the fourth paragraph, add:

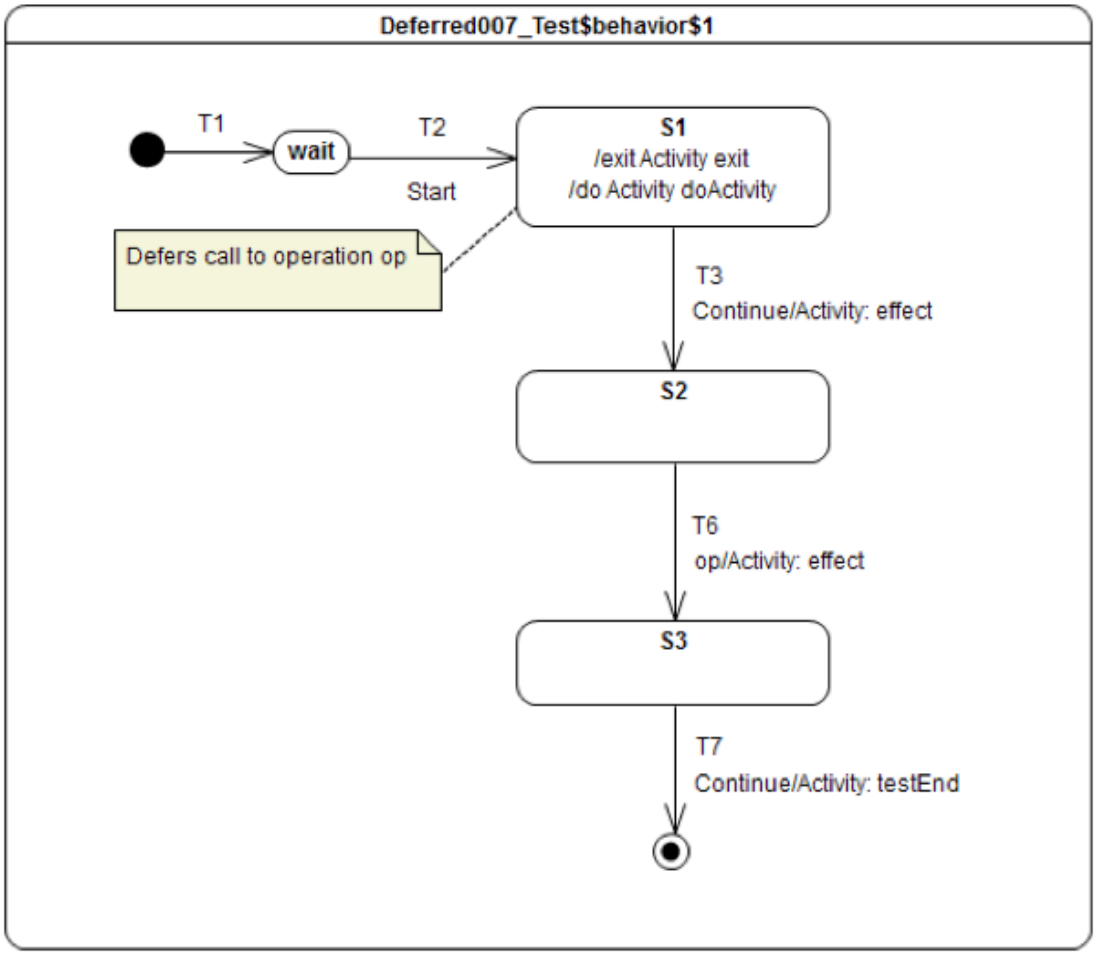
> At the end of the RTC step, if the accepted event was an occurrence of a call event then the object that was responsible for performing the call is released. This implies its thread of execution is resumed.

9.3.16.11 Deferred 007

Under "Tested state machine", add the following sentence after the first sentence:

> The doActivity used in state S1 in Figure 9.101 is shown in Figure 9.102.

Replace the diagram in Figure 9.101 with:



[**Editorial Note.** The above diagram is superseded by the resolution to issue PSSM_-23 - https://issues.omg.org/browse/PSSM_-23.]

After Figure 9.101, add the following new figure (and renumber subsequent figures appropriately):

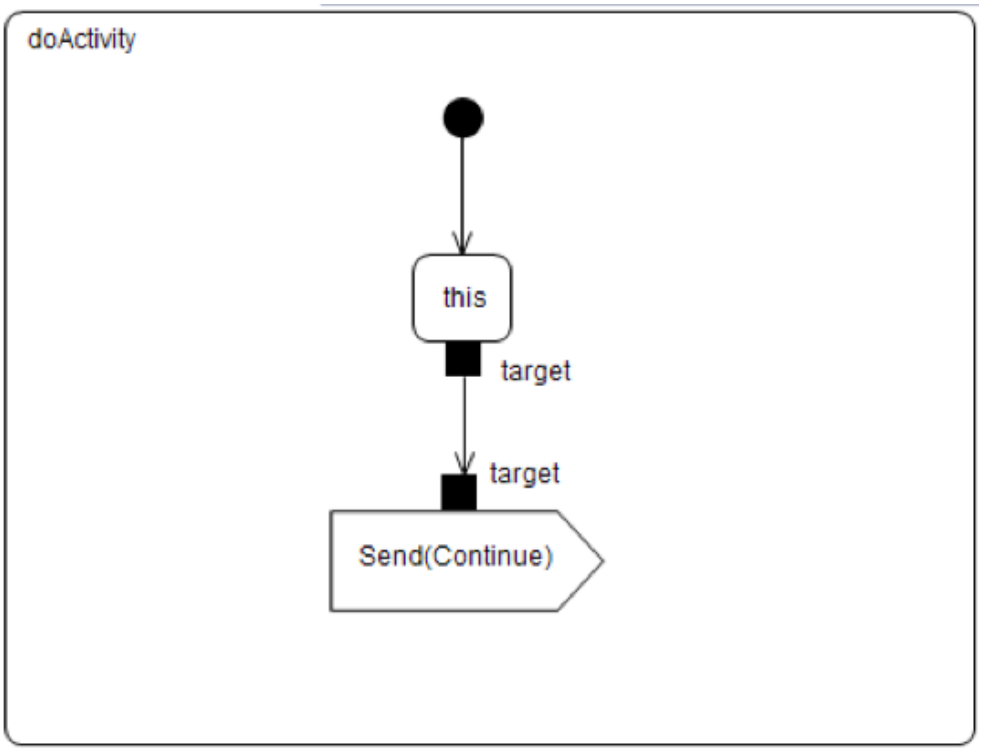


Figure 9.102 Deferred 007 Test doActivity for state S1

Under "Test Execution", update the generated trace to:

> S1(exit)::T3(effect)::T6(effect)[in=true][out=false]::[out=false]

[**Editorial Note.** The above change is superseded by the resolution to issue PSSM_-23 - https://issues.omg.org/browse/PSSM_-23.]

Replace the following note with:

> **Note.** The purpose of this test is to demonstrate that semantics of deferral also applies to call events. Consider the situation where the *Start* event is dispatched and accepted by the state machine. This implies *T2* fires and *S1* is entered. The *doActivity* of *S1* is invoked and will be executed on its own thread of execution. The RTC step of the state machine ends, leaving the state machine in configuration S1. If the call to the operation *op* is received before the *Continue* signal event occurrence sent by the invoked *doActivity*, then the call event occurrence will be deferred by *S1*. This implies that the state machine remains in configuration *S1* at the end of the RTC step. Next, the *Continue* signal event occurrence will be dispatched and accepted. This results in the call event occurrence for *op* being returned to the event pool and transition *T6* being fired. A completion event is generated for *S2*. The completion event is dispatched and lost due to lack of a completion transition outgoing *S2*. The next event to be dispatched is the call event occurrence. It is accepted by the state machine and triggers *T6*. A completion event is generated for *S3* and lost when dispatched. When the *Continue* event occurrence sent by the tester is accepted by the state machine, *T7* fires and the execution completes.
>
> It is important to note that another execution is possible for this test case. Indeed, the *Continue* event occurrence sent by the *doActivity* may be added first to the event pool of the target. In this case, the call event occurrence has not been deferred yet. This means the state machine might be in configuration *S2* at the point the call event occurrence gets accepted by the state machine. As a result, for this particular execution, it is not possible to assess the joint usage of call event and deferral semantics.

[**Editorial Note.** The above change is superseded by the resolution to issue PSSM_-23 - https://issues.omg.org/browse/PSSM_-23.]

Semantics Model

Update the PSSM semantics model, represented in the normative file PSSM_Semantics.xmi, as follows.

class StateMachineAccepter

in the accept operation, replace the initial comment with:

```
// When an event occurrence is accepted this marks the beginning of a new RTC step for
// the executed state-machine. The following set of actions takes place:
// 1 - The event can be deferred if required
// 2 - The event can trigger one or more transitions if it is not deferred
//  2.1 - The list of transitions that can be fired using the given event
//        occurrence is computed.
//  2.2 - Transitions in the set of fireable transitions are fired **concurrently**
//  2.3 - If the accepted event occurrence is a call event occurrence then there is an explicit
//   "return from call" which enables the caller to continue its execution.
// 3 - When the RTC step is about to complete, a new event accepter for the state-machine
//     is registered in the waiting event accepter list handled by the object activation
// Note that there is always just a single event accepter for a state-machine. This accepter
// analyzes the overall state machine configuration each time an event occurrence is accepted.
```

In the accept operation, move the following code:

```
// If the dispatched event was an CallEventOccurrence then check
// if the caller need to be released.
// FIXME: This moved on further updates to common behavior semantics
if(eventOccurrence instanceof CallEventOccurrence){
 CallEventOccurrence callEventOccurrence = (CallEventOccurrence) eventOccurrence;
 callEventOccurrence.execution.releaseCaller();
}
```

to immediately after the for loop nested within the two earlier if statements.

Note: If resolution PSSM_-5 - https://issues.omg.org/browse/PSSM_-5 to issue PSSM_-4 - https://issues.omg.org/browse/PSSM_-4 is accepted, then the above code will be replaced with the following code, which is what should be moved instead:

```
CallEventOccurrence callEventOccurrence = null;
if(eventOccurrence instanceof CS_EventOccurrence){
   EventOccurrence wrappedEventOccurrence = ((CS_EventOccurrence)eventOccurrence).wrappedEventOccurrence;
   if(wrappedEventOccurrence instanceof CallEventOccurrence){
      allEventOccurrence = (CallEventOccurrence) wrappedEventOccurrence;
   }
}else if(eventOccurrence instanceof CallEventOccurrence){
   callEventOccurrence = (CallEventOccurrence) eventOccurrence;
}
if(callEventOccurrence != null){
   callEventOccurrence.returnFromCall();
}
```

Test Suite Model

Update the model for the test Deferred 007, in package Deferred::007 in the PSSM test suite model,

represented in the normative file PSSM_TestSuite.xmi, as follows.

class Deferred007_Test

Update the signature of the operation op from op(in p1: Boolean) to op(in p1: Boolean): Boolean.

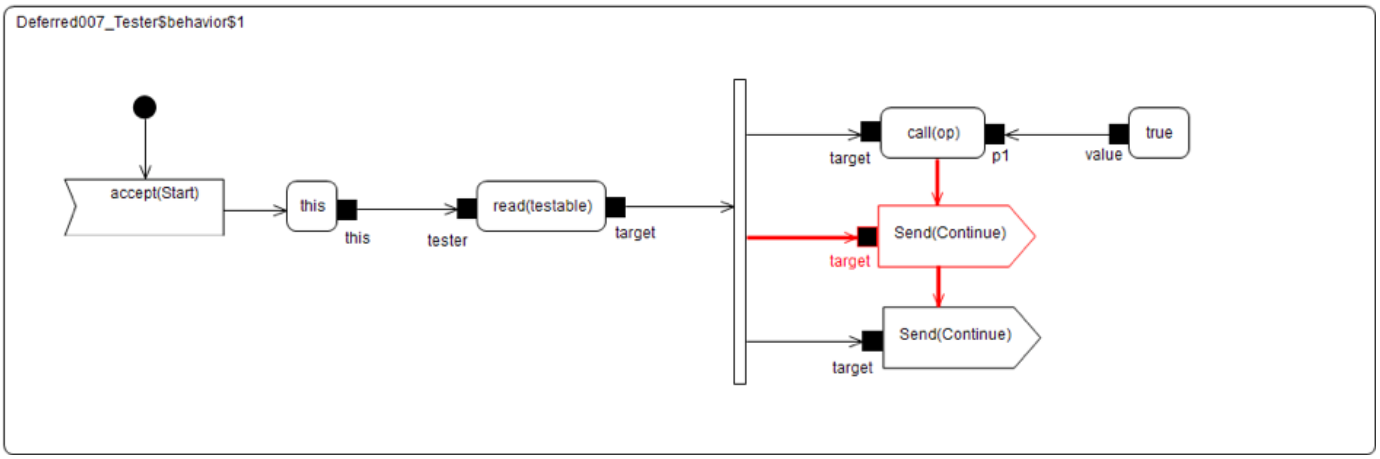
Update the classifier behavior Deferred007_Test$behavior$1 to be consistent with the new diagram given above for Figure 9.101 in the specification document, with the new doActivity for state S1 as given above for Figure 9.102.

Update the effect behavior for transition T6 of the classifier behavior state machine, replacing its Alf specification with:

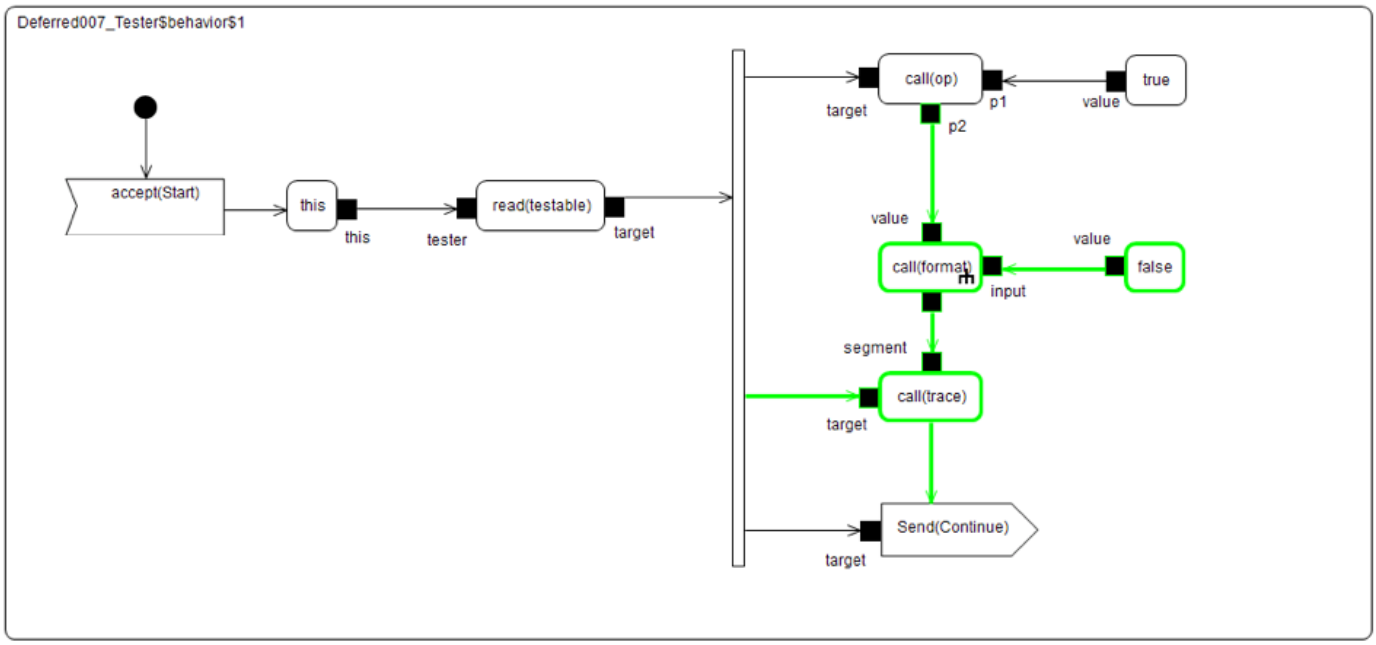
```
activity T6_effect(in p: Boolean) : Boolean {
    this.trace("T6(effect)[in="+BooleanFunctions::ToString(p)+"][out="+BooleanFunctions::ToString(!p)+"]");
    return !p;
}
```

class Deferred007_Tester

Update the classifier behavior Deferred007_Tester$behavior$1 by replacing the Send(Continue) action, shown in red in the diagram below:



with the nodes shown in green below:



activity DeferredTests

In the activity Deferred::DeferredTests, replace the statement

```
d007.expectedTraces->add("S1(exit)::T3(effect)::S2(exit)[in=true]::T6(effect)[in=true]");
```

with

```
d007.expectedTraces->add("S1(exit)::T3(effect)::T6(effect)[in=true][out=false]::[out=false]");
```

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #2
Reporter:
Jeremie Tatibouet, Commissariat a l Energie Atomique-CEA (jeremie.tatibouet@cea.fr)
Reported:
Mon, 15 May 2017 15:47 GMT on PSSM 1.0b1
Updated:
Fri, 9 Nov 2018 21:20 GMT
Discussion:
PSSM_-2 - https://issues.omg.org/browse/PSSM_-2
Attachments:

- Deferred007_Test.PNG - ptc-18-11-07/PSSM_-2/Deferred007_Test.PNG 14 kB ()
- Figure-9.101.png - ptc-18-11-07/PSSM_-2/Figure-9.101.png 47 kB (image/png)
- Figure-9.102.png - ptc-18-11-07/PSSM_-2/Figure-9.102.png 20 kB (image/png)
- New-Tester-Behavior.png - ptc-18-11-07/PSSM_-2/New-Tester-Behavior.png 47 kB (image/png)
- Old-Tester-Behavior.png - ptc-18-11-07/PSSM_-2/Old-Tester-Behavior.png 37 kB (image/png)

OMG Issue: PSSM_-4

Title:

PSSM shall be aligned with fUML 1.3 and PSCS 1.1

Summary:

PSSM is not compatible with fUML 1.3 and PSCS 1.1. This is the consequence of the changes introduced by FUML13-23, FUML13-25, FUML13-16, FUML13-1, FUML13-60 and PSCS11-6. The description below provides an overview of the changes that must be performed in the PSSM semantic model in order to make PSSM comptabile with fUML 1.3 and PSCS 1.1. Note that these changes will also require an update of the PSSM document.

Source:

ad/2016-11-01

Resolution Summary:

Update PSSM for fUML 1.3

There are four categories of changes needed to align PSSM with fUML 1.3 and PSCS 1.1:

1. *Changes to the signature of the Object::send operation (see [fUML 1.3], 8.3.2.2.19, and issue resolution FUML13-23 - https://issues.omg.org/browse/FUML13-23)*
   This requires adjusting the implementation of DoActivityContextObject, which redefines the inherited send operation.
2. *API changes introduced in the EventOccurrence class (see issue resolutions FUML13-25 - https://issues.omg.org/browse/FUML13-25, FUML13-1 - https://issues.omg.org/browse/FUML13-1, and FUML13-60 - https://issues.omg.org/browse/FUML13-60)*
   These API changes impact the specification of the trigger-matching semantics of CompletionEventOccurrence and DeferredEventOccurrence (see [PSSM 1.0b], 8.5.9). Specifically, a completion event cannot match a trigger since it is used to only trigger transitions with no triggers, and a deferred event occurrence merely delegates to the matching semantics of the deferred event. fUML now also provides the capability for an EventOccurrence to be sent through its own execution thread, but neither the CompletionEventOccurrence nor the DeferredEventOccurrence use this capability. Finally, the alignment with the new EventOccurrence API enables the refactoring of the code of StateActivation (canDefer, defer and notifyCompletion operations), SM_ObjectActivation (registerCompletionEvent operation), TransitionActivation (canFireOne and hasTrigger operations), StateMachineEventAccepter (isDeferred operation) and StateMachineSemanticVisitor (removal of the match operation).
3. *Introduction of class CS_EventOccurrence in PSCS (see [PSSM 1.0b], 8.5.1.2.7, and issue resolution PSCS11-6 - https://issues.omg.org/browse/PSCS11-6)*
   This requires refactoring EventTriggeredExecution (see [PSSM 1.0b], 8.5.10.1), StateMachineEventAccepter (see [PSSM 1.0b], 8.5.2) and SM_OpaqueExpressionEvaluation (see [PSSM 1.0b], 8.2) to account for the possibility of receiving an CS_EventOccurrence .
4. *Handling of call events in fUML (see [fUML 1.3], 7.3.3 and 8.8, and issue resolution FUML13-16 - https://issues.omg.org/browse/FUML13-16)*
   Since fUML 1.3 now includes CallEvent, this no longer needs to be added in the syntax subset for PSSM (see [PSSM 1.0b], 7.5), and CallEventOccurrence and CallEventExecution (and all the associations in which they are involved) can be removed from the semantics for PSSM (see [PSSM 1.0b], 8.5.9).

Revised Text:

Make the following updates to the specification document and, the syntax model (as represented in the normative PSSM Syntax XMI), and the semantic model (as represented in the normative PSSM Semantics XMI).

3 Normative References

In the reference for "[fUML]", replace "Version 1.2.1" with "Version 1.3".

In the reference for "[PSCS]", replace "version 1.0" with "Version 1.1".

[**Editorial Note.** The above changes are superseded by the resolution to PSSM_-12 - https://issues.omg.org/browse/PSSM_-12.]

6 Additional Information

6.2 Changes to Adopted OMG Specifications

In the second sentence, change "fUML 1.2.1" to "fUML 1.3" and "PSCS 1.0" to "PSCS 1.1".

[**Editorial Note.** The above changes are superseded by the resolution to PSSM_-12 - https://issues.omg.org/browse/PSSM_-12.]

7 Abstract Syntax

7.1 Overview

In Figure 7.1 and in the syntax model, make the following changes:

1. Remove package PSSM_Syntax::Syntax::CommonBehavior.
2. Add a package import relationship from package PSSM_Syntax to package fUML_Syntax::Syntax::CommonBehavior.

In Figure 7.2 and in the syntax model, make the following changes:

1. Remove package PSSM_Syntax::Constraints::CommonBehavior (and, in the syntax model, the constraint it contains).
2. Add a package import relationship from package PSSM_Syntax to package fUML_Syntax::Constraints::CommonBehavior.

[**Editorial Note.** The text of 7.1 also needs to be updated to reflect the changes to the figures and the removal of subclause 7.5 (see below).]

7.5 Common Behavior

Remove this subclause and renumber subsequent subclauses appropriately.

8 Execution Model

8.2 Values

In the semantic model, make the following changes to class SM_OpaqueExpressionEvaluation.

class Values::SM_OpaqueExpressionEvaluation

In the implementation of operation initialize, before

```
this.parameterValues.clear()
```

add

```
EventOccurrence currentEventOccurrence = eventOccurrence;
if(eventOccurrence instanceof CS_EventOccurrence){
 currentEventOccurrence = ((CS_EventOccurrence)eventOccurrence).wrappedEventOccurrence;
}
```

In the rest of the code, replace every occurrence of eventOccurrence with currentEventOccurrence.

Also replace

```
callEventOccurrence.getParameterValues();
```

with

```
callEventOccurrence.execution.getInputParameterValues();
```

8.4 Common Behavior

In Figure 8.4 and in the semantic model, in class SM_ObjectActivation, change

```
+ registerCompletionEvent(stateActivation: StateActivation[1])
```

to

```
+ register(completionEventOccurrence: CompletionEventOccurence[1]).
```

In addition, in the semantic model, make the following changes to class SM_ObjectActivation.

class CommonBehavior::SM_ObjectActivation

In the implementation of operation register, remove the following statements:

```
CompletionEventOccurrence completionEventOccurrence = new CompletionEventOccurrence();
completionEventOccurrence.stateActivation = stateActivation;
```

[**Editorial Note.** In addition, the start of the first paragraph of 8.4 needs to be updated to reflect the removal of subclause 7.5 (see above).]

8.5 State Machines

8.5.2 State Machine Execution

In the semantic model, make the following changes to class StateMachineEventAccepter.

class StateMachines::StateMachineEventAccepter

In the implementation of operation accept, replace:

```
// If the dispatched event was an CallEventOccurrence then check
// if the caller need to be released.
// FIXME: This moved on further updates to common behavior semantics
if(eventOccurrence instanceof CallEventOccurrence){
 CallEventOccurrence callEventOccurrence = (CallEventOccurrence) eventOccurrence;
 callEventOccurrence.returnFromCall();
}
```

with

```
CallEventOccurrence callEventOccurrence = null;
if(eventOccurrence instanceof CS_EventOccurrence){
 EventOccurrence wrappedEventOccurrence = ((CS_EventOccurrence)eventOccurrence).wrappedEventOccurrence;
 if(wrappedEventOccurrence instanceof CallEventOccurrence){
  callEventOccurrence = (CallEventOccurrence) wrappedEventOccurrence;
 }
}else if(eventOccurrence instanceof CallEventOccurrence){
 callEventOccurrence = (CallEventOccurrence) eventOccurrence;
}
if(callEventOccurrence != null){
 callEventOccurrence.returnFromCall();
}
```

In the implementation of operation isDeferred, replace

```
boolean deferred = this._isDeferred(eventOccurrence, this.registrationContext.getConfiguration().rootConfiguration);
```

with

```
// Note: a completion event cannot be deferred.
boolean deferred = false;
if(!(eventOccurrence instanceof CompletionEventOccurrence)){
 deferred = this._isDeferred(eventOccurrence, this.registrationContext.getConfiguration().rootConfiguration);
}
```

8.5.3 State Machine Semantic Visitors

In Figure 8.6, in class StateMachineSemanticVisitor, remove the match operation. Also remove this operation in the semantic model, along with its implementation.

8.5.5 State Activations

In the semantic model, make the following changes to class StateActivation.

class StateMachines::StateActivation

In the implementation of operation canDefer, replace

```
this.match(eventOccurrence, state.getDeferrableTriggers());
```

with

```
eventOccurrence.matchAny(state.getDeferrableTriggers());
```

Replace the implementation of operation notifyCompletion with:

```
// StateActivation completion consists in sending in the execution
// context of the state-machine a completion event occurrence. This event is
// placed in the pool before any other event
CompletionEventOccurrence completionEventOccurrence = new CompletionEventOccurrence();
completionEventOccurrence.register(this);
```

Replace the implementation of operation defer with:

```
// Postpone the time at which this event occurrence will be available at the event pool.
// The given event occurrence is placed in the deferred event pool and will be released
// only when the current state activation will leave the state-machine configuration.
DeferredEventOccurrence deferringEventOccurrence = new DeferredEventOccurrence();
deferringEventOccurrence.deferredEventOccurrence = eventOccurrence;
deferringEventOccurrence.register(this);
```

8.5.6 "doActivity" Behavior Execution

In Figure 8.9 and in the semantic model, update the the type of the input parameter of the DoActivityContextObject::send operation from SignalEvent to EventOccurrence.

In addition, in the semantic model, make the following change to class DoActivityObjectActivation.

class StateMachines::DoActivityObjectActivation

In the implementation of operation send, replace

```
this.context.send(signalInstance);
```

with

```
this.context.send(eventOccurrence);
```

8.5.8 Transition Activations

In the semantic model, make the following changes to class TransitionActivation.

class StateMachines::TransitionActivation

In the implementation of operation hasTrigger, replace

```
this.match(eventOccurrence, transition.getTriggers());
```

with

```
eventOccurrence.matchAny(transition.getTriggers());
```

Replace implementation of operation canFireOn with:

```
// A transition is can fire when:
// 1. It has a trigger that matches the dispatched event occurrence.
// 2. Its guard evaluates to true.
// 3. A valid path can found to the next state machine configuration.
// Note: If the dispatched event is a completion event, the transition matches this latter
// if it has no trigger and the transition leaves the state from which the completion event
// was generated.
boolean reactive = this.hasTrigger(eventOccurrence) &&
      this.evaluateGuard(eventOccurrence) &&
      this.canPropagateExecution(eventOccurrence);
if(eventOccurrence instanceof CompletionEventOccurrence){
 reactive = this.getSourceActivation()==((CompletionEventOccurrence)eventOccurrence).stateActivation;
}
return reactive;
```

8.5.9 Event Occurrences

In the first paragraph, at the end of the third sentence, replace "as is the SignalEventOccurrence class" with "as are the SignalEventOccurrence and CallEventOccurrence classes". Remove the subsequent sections under the headings CallEventOccurrence and CallEventExecution.

In Figure 8.15 and in the semantic model, make the following changes:

1. Remove CallEventOccurrence class.
2. Remove CallEventExecution class.
3. Remove association A_operation_callEventExecution.
4. Remove association A_behavior_callEventExecution.
5. Remove association A_callerContext_callEventExecution.
6. Remove association A_execution_callEventOccurrence.
7. Add the following operations to class CompletionEventOccurrence:

```
+ match (in trigger: Trigger[1]) : Boolean
+ match (in triggers: Trigger[0..*]) : Boolean
+ getParameterValues(): ParameterValue[0..*]
+ sendTo(target: Reference)
+ doSend()
+ register(stateActivation: StateActivation[1])
```

8. Add the following operations to class DeferredEventOccurrence:

```
+ match (in trigger: Trigger[1]) : Boolean
+ getParameterValues(): ParameterValue[0..*]
+ sendTo(target: Reference)
+ doSend()
+ register(stateActivation: StateActivation[1])
```

In addition, in the semantic model, add the following operation implementations to classes CompleteEventOccurrence and DeferredEventOccurrence.

class StateMachines::CompletionEventOccurrence

match (in trigger: Trigger[1]) : Boolean

```
// A completion event can only trigger transition with no
// trigger. Hence it cannot match a trigger so false is returned
// regardless the trigger provided as parameter.
return false;
```

match (in triggers: Trigger[0..*]) : Boolean

```
// A completion event can only trigger transition with no
// trigger. Hence if the list of trigger that is passed is
// empty then the completion event occurrence matches. It does match
// otherwise.
boolean match = false;
if(triggers.size() == 0){
 match = true;
}
return match;
```

getParameterValues(): ParameterValue[0..*]

```
// A completion will never have associated values. Hence a
// empty list is returned.
return new ArrayList<ParameterValue>();
```

sendTo(target: Reference)

```
// Do nothing - the completion event is not sent to a target.
// It is registered during the RTC step of the active object that
// entered the state from which it was generated.
```

doSend()

```
// Do nothing - the completion event is not sent to a target.
// It is registered during the RTC step of the active object that
// entered the state from which it was generated.
```

register(stateActivation: StateActivation[1])

```
// Register this completion event occurrence in the event
// pool of the context object.
this.stateActivation = stateActivation;
SM_ObjectActivation objectActivation = (SM_ObjectActivation)stateActivation.getExecutionContext().objectActivation;
objectActivation.register(this);
```

class StateMachines::DeferredEventOccurrence

match(trigger: Trigger[1]) : Boolean

```
boolean match = false;
if(this.deferredEventOccurrence != null){
 match = this.deferredEventOccurrence.match(trigger);
}
return match;
```

getParameterValues(): ParameterValue[0..*]

```
// Delegate to the getParameterValues operation of the encapsulated event
// occurrence which is the one being deferred.
List<ParameterValue> parameterValues = new ArrayList<ParameterValue>();
if(this.deferredEventOccurrence != null){
 parameterValues = this.deferredEventOccurrence.getParameterValues();
}
return parameterValues;
```

sendTo(target: Reference)

```
// Do nothing - the deferred event is not sent to a target.
// It is registered during the RTC step of the active object that
// entered the state from which it was generated.
```

doSend()

```
// Do nothing - the deferred event is not sent to a target.
// It is registered during the RTC step of the active object that
// entered the state from which it was generated.
```

register(stateActivation: StateActivation[1])

```
// Register this deferred event occurrence in the deferred
// event pool of the context object.
this.constrainingStateActivation = stateActivation;
SM_ObjectActivation objectActivation = (SM_ObjectActivation)stateActivation.getExecutionContext().objectActivation;
objectActivation.register(this);
```

8.5.10.1 Event Triggered Execution

In the semantic model, make the following changes to the class EventTriggeredExecution.

class CommonBehaviors::EventTriggeredExecution

In the implementation of operation initialize, immediately after

```
Behavior behavior = this.wrappedExecution.getBehavior();
```

add the following statements:

```
EventOccurrence currentEventOccurrence = this.triggeringEventOccurrence;
if(this.triggeringEventOccurrence instanceof CS_EventOccurrence){
 currentEventOccurrence = ((CS_EventOccurrence)this.triggeringEventOccurrence).wrappedEventOccurrence;
}
```

In the rest of the code, replace every occurrence of this.triggeringEventOccurrence with currentEventOccurrence.

In the implementation of operation finalize, immediately after

```
this._beginIsolation();
```

add the following statements:

```
EventOccurrence currentEventOccurrence = this.triggeringEventOccurrence;
if(this.triggeringEventOccurrence instanceof CS_EventOccurrence){
 currentEventOccurrence = ((CS_EventOccurrence)this.triggeringEventOccurrence).wrappedEventOccurrence;
}
```

In the rest of the code, replace every occurrence of this.triggeringEventOccurrence with currentEventOccurrence.

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #2
Reporter:
Jeremie Tatibouet, Commissariat a l Energie Atomique-CEA (jeremie.tatibouet@cea.fr)
Reported:
Thu, 13 Apr 2017 12:24 GMT on PSSM 1.0b1
Updated:
Fri, 9 Nov 2018 17:02 GMT
Discussion:
PSSM_-4 - https://issues.omg.org/browse/PSSM_-4

OMG Issue: PSSM_-7

Title:

In Testcase "Entering 011" T3 is not a completion transition

Summary:

The text says:

> The S1 completion event is then used to trigger transition T3.

Actually transition T3 has an event "Continue" in the diagram and will not react to a completion event.

Source:
ptc/2017-04-01 — Chapter/Section: 9.3.5.6 — Page Number/s: 130

Resolution Summary:

Resolution of issue PSSM-7

Entering 011 has a transition T3 that is not a completion transition. The text describing the interpretation of this transition is inconsistent since it considers T3 has a completion transition.

This issue is only related to the text. Indeed the table describing each RTC step performed during the execution of the test is consistent with the model (i.e., it consider T3 as a transition triggered by the acceptance of a Continue event occurrence).

Revised Text:

Specification Document

In 9.3.5.6 Entering 011, in the note under the heading **Generated Trace**

Replace

> The S1 completion event is then used to trigger transition T3.

with

> At this point, the completion event generated by S1 is dispatched and lost since T3 is not a completion transition. The acceptance of the Continue event occurrence causes T3 to be traversed which leads the test to end.

This issue resolution has no impact on the normative XMI

Extent Of Change:
Minor

Disposition:
Resolved - Approved on Ballot #2

Reporter:
Axel Scheithauer, oose Innovative Informatik eG (axel.scheithauer@oose.de)

Reported:
Tue, 3 Apr 2018 15:35 GMT on PSSM 1.0b1

Updated:
Fri, 9 Nov 2018 00:05 GMT

Discussion:
PSSM_-7 - https://issues.omg.org/browse/PSSM_-7

OMG Issue: PSSM_-8

Title:

"Join 003" test case state machine diagram appears to be invalid

Summary:

According to the UML 2.5.1 spec, join pseudostates have a constraint `join_vertex` that ensures that joins have exactly one outgoing transition. Similarly, fork pseudostates have a constraint `fork_vertex` that ensures that forks have exactly one incoming transition. The state machine in "Join 003" includes a heavy bar (which could be interpreted as either a fork or join) that has two incoming transitions and two outgoing transitions. But due to the constraints, it can't be either. In the notation section of the UML 2.5.1 spec, there doesn't seem to be any affordance for a single drawn symbol in the state machine diagram to represent both a fork AND a join instance from the metamodel.

This diagram appears to violate the `join_vertex` constraint of the metamodel.

Source:

ptc/17-04-04 — Chapter/Section: 9.3.12.4 Join 003 — Page Number/s: 179

Resolution Summary:

Resolution of issues PSSM_-8 - https://issues.omg.org/browse/PSSM_-8 and PSSM_-11 - https://issues.omg.org/browse/PSSM_-11

Agreed. **Join003_Test** violates constraints join_segment_guards (see subclause 14.5.11.8 in [UML 2.5.1]) and join_vertex (see subclause 14.5.6.7 in [UML 2.5.1]).

- join_segment_guards implies that a transition targeting a join pseudostate cannot have a guard and trigger.
- join_vertex implies that a join pseudo state can only have a single outgoing transition.

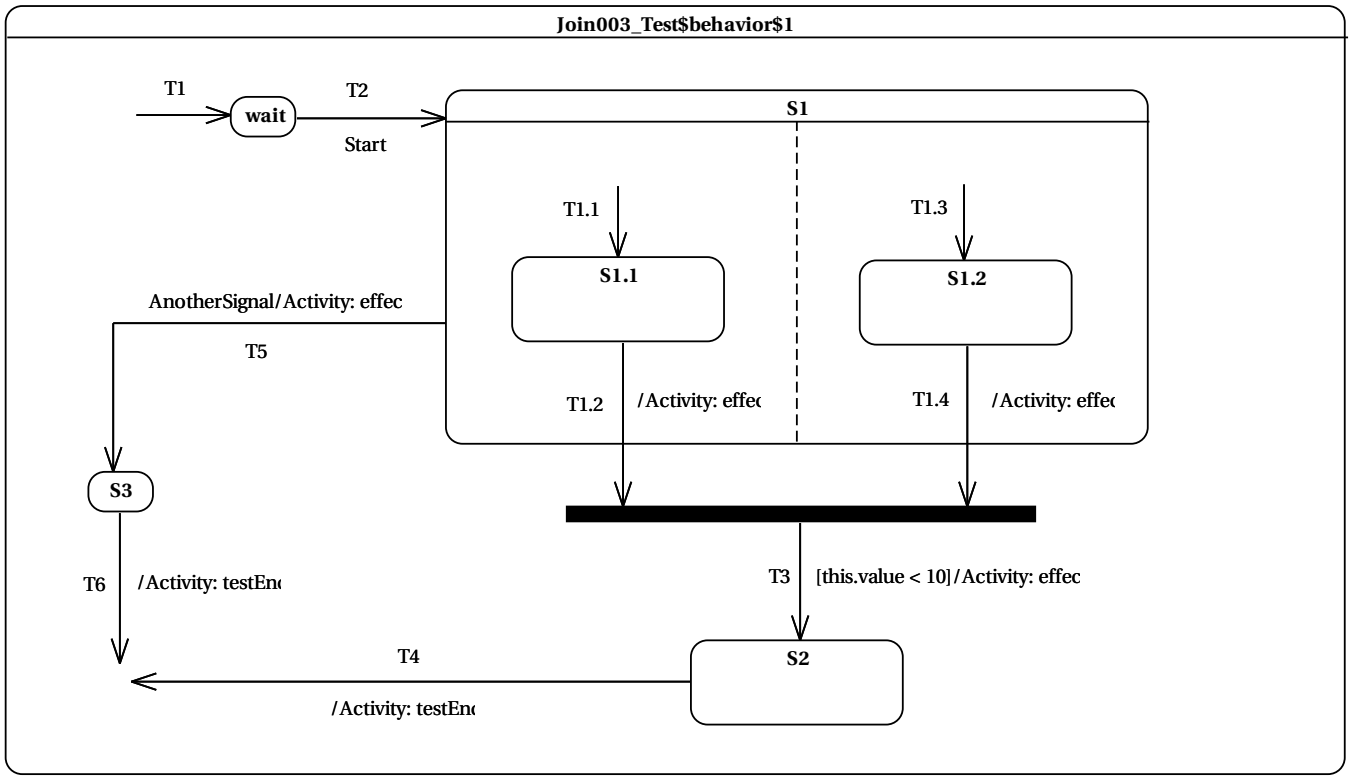
Hence, the state machine specifying the test behavior shall be updated as well as the description provided in the subclause 9.3.12.4 of [PSSM 1.0b].

Revised Text:

Specification Document

Apply the following changes to subclause 9.3.12.4 in [PSSM 1.0b]:

Replace Figure 9.78 with



Under the heading **Received event occurrence(s)**

Replace

IntegerData(8) - received in configuration wait

with

AnotherSignal - received either :

- in configuration S1[S1.1]
- in configuration S1[S1.2]

Under the heading **Generated trace**

Replace

T2(effect)::T1.4(effect)[in=8]::T3(effect)[in=8]

with

T1.2(effect)::T5(effect)

Next, the replace the entire **Note** with

The test is focused on the application of the static analysis when a join pseudostate is encountered in a compound transition. The feature value is assigned to 15 during test initialization. When Start event occurrence is accepted by the state machine, it reaches the configuration S1[S1.1, S1.2]. As both S1.1 and S1.2 complete two completion events are generated and placed into the event pool. Under the assumption that the completion event for S1.1 is dispatched and accepted first, the state machine moves to the configuration S1[S1.2]. Next, the completion event for S1.2 is dispatched. However, it cannot be accepted by the state machine. Indeed, at static analysis time, the compound transition T1.4(T3) cannot be used to reach a valid state machine configuration. This situation is rationalized

by the guard on T3 which prevents the execution to reach S2. Hence, the completion event for S1.2 is lost. The next event occurrence available at the pool is AnotherSignal. When accepted by the state machine, T5 is traversed and the configuration S3 is reached. The completion event for S3 is used to trigger T6 which leads the test to complete its execution.

Under the heading **RCT Steps**

Replace the first table (i.e., right below the title) with

Step	Event pool	State machine configuration	Fired transition(s)
1	[]	[] - Initial RTC Step	[T1]
2	[Start, **CE(wait)**]	[wait]	[]
3	[**Start**]	[wait]	[T2(T1.1, T1.3)]
4	[AnotherSignal, CE(S1.2), **CE(S1.1)**]	[S1[S1.1, S1.2]]	[T1.2]
5	[AnotherSignal, **CE(S1.2)**]	[S1[S1.2]]	[]
6	[**AnotherSignal**]	[S1[S1.2]]	[T5]
7	[**CE(S3)**]	[S3]	[T6]

Next, replace the three alternative execution traces with

T1.4(effect)::T5(effect)

Finally, replace the second table with

Step	Event pool	State machine configuration	Fired transition(s)
1	[]	[] - Initial RTC Step	[T1]
2	[Start, **CE(wait)**]	[wait]	[]
3	[**Start**]	[wait]	[T2(T1.1, T1.3)]
4	[AnotherSignal, CE(S1.1), **CE(S1.2)**]	[S1[S1.1, S1.2]]	[T1.4]
5	[AnotherSignal, **CE(S1.1)**]	[S1[S1.1]]	[]
6	[**AnotherSignal**]	[S1[S1.1]]	[T5]
7	[**CE(S3)**]	[S3]	[T6]

Test Suite Model

Update the PSSM test suite XMI (i.e., PSSM_Tests.xmi) according to the changes introduced in above state machine.

Semantic Model

Update the PSSM semantics model (i.e., PSSM_Semantics.xmi) as follows.

class JoinPseudostateActivation

In the canPropagateExecution operation, replace the content of the main if statement with :

JoinPseudostateActivation.java

```
this.evaluateAllGuards(eventOccurrence);
```

```
propagate = false;
if(this.fireableTransitions.size() > 0){
 propagate = this.fireableTransitions.get(0).canPropagateExecution(eventOccurrence);
}
if(propagate) {
 this.tagIncomingTransitions(TransitionMetadata.NONE, true);
}
```

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #3
Reporter:
Daniel Yankowsky, AGI (dyankowsky@agi.com)
Reported:
Wed, 29 Aug 2018 19:22 GMT on PSSM 1.0b1
Updated:
Mon, 12 Nov 2018 00:16 GMT
Discussion:
PSSM_-8 - https://issues.omg.org/browse/PSSM_-8
Attachments:

- Join003_Test-v1.1.svg - ptc-18-11-07/PSSM_-8/Join003_Test-v1.1.svg 36 kB (image/svg+xml)

OMG Issue: PSSM_-9

Title:

Example state machine appears to violate UML constraints

Summary:

In this state machine, there is a fork pseudostate with two outgoing transitions. One transition targets a state, while the other transition appears to target a junction pseudostate.

The UML 2.5.1 spec includes the constraint `fork_segment_state` (14.5.11.8 Constraints) that requires that any transition originating at a fork must terminate at a state. Such a transition is not allowed to terminate at a pseudostate.

(Incidentally, that particular constraint doesn't seem particularly useful, as any undesirable case that the constraint is trying to prevent could be instead created via default entry into an orthogonal state. But the constraint exists nonetheless.)

Source:

ptc/17-04-04 — Chapter/Section: 9.3.3.15 Transition 023 — Page Number/s: 99

Resolution Summary:

Resolution of issue PSSM_-9 - https://issues.omg.org/browse/PSSM_-9

Agreed. **Transition_023** violates constraint fork_segment_state (see subclause 14.5.11.8 in [UML 2.5.1]). Indeed, transition T4 outgoing the fork pseudostate directly targets a junction pseudostate. This is forbidden since the constraints specifies that a transition outgoing a fork pseudostate shall only have a state as a target.

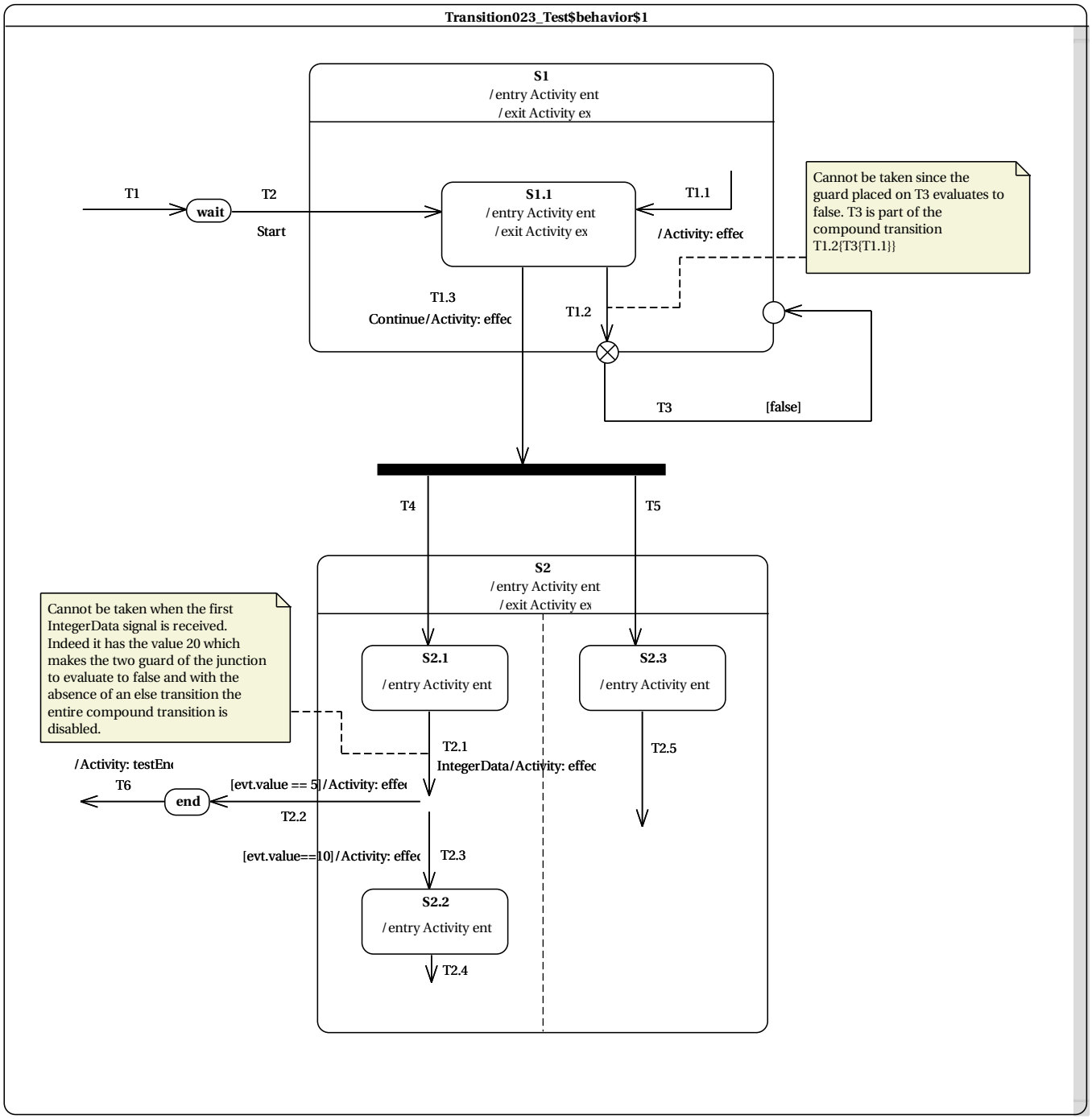
Hence, the state machine specifying the test behavior shall be updated as well as the description provided in the subclause 9.3.3.15 of [PSSM 1.0b].

Revised Text:

Specification Document

Apply the following changes to subclause 9.3.3.15 in [PSSM 1.0b]

Replace Figure 9.25 with



Under the heading **Received event occurrence(s)**

Add the following bullet point right after the one dedicated to the Start event occurrence.

- Continue - received in configuration S1[S1.1]

Next, replace bullet points dedicated to the IntegerData event occurrence with

- IntegerData – received when in configuration S2[S2.1, S2.3]. The property value has the value 20.
- IntegerData – received when in configuration S2[S2.1, S2.3]. The property value has the value 5.

Under the heading **Generated trace**

Replace

S1(entry)::S1.1(entry)::S1.1(exit)::S1(exit)::T1.3(effect)[in=5]::S2(entry)[in=5]::S2.2(entry)[in=5]::S2(exit)[in=5]::T2.1(effect)[in=5]

with

S1(entry)::S1.1(entry)::S1.1(exit)::S1(exit)::T1.3(effect)::S2(entry)::S2.1(entry)::S2.3(entry)::T2.1(effect)[in=5]::S2(exit)::T2.2(effect)[in=5]

Next, replace all paragraphs of the **Note** except the first one with

In this test case, the state machine receives four event occurrences: Start, Continue, IntegerData(20) and IntegerData(5). The dispatching of the Start event brings the state-machine to the configuration S1[S1.1] due to the triggering of T2. The completion event generated for S1.1 is dispatched and lost since it cannot bring the state machine to a valid configuration. Indeed event if the completion event could have been used to trigger the compound transition starting with T1.2, T3 could not have been traversed since its guard evaluated to false.

Next, the Continue event occurrence is dispatched and accepted by the state machine. When the RTC step ends, the configuration is S2[S2.1, S2.3]. It will remain the same until IntegerData(5) is dispatched. This can be explained because the received IntegerData(20) signal event cannot bring the state machine to a valid configuration. Indeed even if the event occurrence could have been used to trigger the compound transition starting with T2.1 neither T2.2 nor T2.3 could have been traversed since their guards both evaluate to false (because of the value associated with the signal event occurrence). Hence IntegerData(20) is lost.

When IntegerData(5) is finally dispatched it triggers the traversal of the compound transition T2.1(T2.2).

Under the heading **RCT Steps**

Replace the table with

Step	**Event pool**	**State machine configuration**	**Fired transition(s)**
1	[]	[] - Initial RTC Step	[T1]
2	[Start, **CE(wait)**]	[wait]	[]
3	[**Start**]	[wait]	[T2]
4	[Continue, **CE(S1.1)**]	[S1[S1.1]]	[]
5	[**Continue**]	[S1[S1.1]]	[T1.3[T4, T5]]
6	[IntegerData(5), IntegerData(20), CE(2.3), **CE(S2.1)**]	[S2[S2.1, S2.3]]	[]
7	[IntegerData(5), IntegerData(20), **CE(2.3)**]	[S2[S2.1, S2.3]]	[T2.5]
8	[IntegerData(5), **IntegerData(20)**]	[S2[S2.1]]	[]
9	[**IntegerData(5)**]	[S2[S2.1]]	[T2.2]
10	[**CE(end)**]	[end]	[T6]

Under the heading **Alternative execution traces**

Replace

S1(entry)::S1.1(entry)::S1.1(exit)::S1(exit)::T1.3(effect)[in=5]::S2(entry)[in=5]::S2(exit)

[in=5]::T2.1(effect)[in=5]

with

S1(entry)::S1.1(entry)::S1.1(exit)::S1(exit)::T1.3(effect)::S2(entry)::S2.3(entry)::S2.1(entry)::T2.1(effect)[in=5]::S2(exit)::T2.2(effect)[in=5]

Replace the last paragraph

The difference from the previous trace is that the entry behavior of S2.3 might be executed before S2.1 entry behavior. This is may occur when the compound transition T1.3[T4, T5] is executed.

Test Suite Model

Update the PSSM test suite XMI (i.e., PSSM_Tests.xmi) according to the changes introduced in above state machine.

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #3
Reporter:
Daniel Yankowsky, AGI (dyankowsky@agi.com)
Reported:
Thu, 13 Sep 2018 17:26 GMT on PSSM 1.0b1
Updated:
Mon, 12 Nov 2018 00:16 GMT
Discussion:
PSSM_-9 - https://issues.omg.org/browse/PSSM_-9
Attachments:

- Transition023_Test-v1.0.svg - ptc-18-11-07/PSSM_-9/Transition023_Test-v1.0.svg 64 kB (image/svg+xml)

OMG Issue: PSSM_-10

Title:

Typos in "Note" section of "9.3.3.12 Transition 019"

Summary:

The "Note" section says "Consider the situation where the state machine is in configuration S1[S1.1, S1.2]". This is an impossible configuration; it should read "... in configuration S1[S1.1, S2.1]". The note later says "CE(2.1) will be triggered next", but it should read "CE(2.2) will be triggered next".

Source:

ptc/17-04-04 — Chapter/Section: 9.3.3.12 Transition 019 — Page Number/s: 94

Resolution Summary:

Resolution of issue PSSM_-10 - https://issues.omg.org/browse/PSSM_-10

This is correct.

- At the third line of the note, S1[S1.1, S1.2] designates an invalid configuration. The intended configuration is S1[S1.1, S2.1].
- At the last line of the note, the reference to the completion event for S2.1 is invalid. Indeed, this reference must be for the completion event generated for S2.2.

The changes to perform are limited to the **Note** in subclause 9.3.3.12 of [PSSM 1.0b].

Revised Text:

Specification Document

Under the heading **Generated trace** of subclause 9.3.3.12

Replace S1[S1.1, S1.2] at the third line of the **Note** with

S1[S1.1, S2.1]

Replace CE(2.1) at the last line of the **Note** with

CE(S2.2)

This issue resolution has no impact on the normative XMI

Extent Of Change:

Minor

Disposition:

Resolved - Approved on Ballot #2

Reporter:

Daniel Yankowsky, AGI (dyankowsky@agi.com)

Reported:

Thu, 13 Sep 2018 18:33 GMT on PSSM 1.0b1

Updated:

Fri, 9 Nov 2018 17:27 GMT

Discussion:

PSSM_-10 - https://issues.omg.org/browse/PSSM_-10

OMG Issue: PSSM_-12

Title:

PSSM should align with fUML 1.4 and PSCS 1.2

Summary:

The fUML 1.4 and PSCS 1.2 specifications have now been completed. These updates make no functional changes to fUML or PSCS, but they migrate those standards to UML 2.5.1, which is consistent with PSSM. Therefore, PSSM should align with fUML 1.4 and PSCS 1.2, rather than fUML 1.3 and PSCS 1.1.

Resolution Summary:

Update specification for fUML 1.4 and PSCS 1.2

Since fUML 1.4 and PSCS 1.2 are based on UML 2.5.1, it is no longer necessary for the PSSM specification to include fUML and PSCS syntax and semantics files based on UML 2.5.1. Instead, the PSSM syntax, semantics and test suite should just use UML 2.5.1, fUML 1.4 and PSCS 1.2.

The normative references in the specification document need to be updated to fUML 1.4 and PSCS 1.2. In addition, since both the fUML and the PSCS specification documents are now re-organized to follow UML 2.5.1, any subclause references to the fUML and PSCS documents need to be updated.

Revised Text:

Note: Unless otherwise noted below, this issue resolution presumes the adoption of all the changes proposed in resolution PSSM_-5 - https://issues.omg.org/browse/PSSM_-5 to issue PSSM_-4 - https://issues.omg.org/browse/PSSM_-4, because the functional changes required to align with fUML 1.3 all still apply for aligning with fUML 1.4.

Normative XMI

- Remove the PSSM-specific normative XMI files fUML_Syntax.xmi, fUML_Semantics.xmi, PSCS_Syntax.xmi and PSCS_Semantics.xmi.
- Update PSSM_Syntax.xmi to reference the fUML 1.4 and PSCS 1.2 syntax models, as appropriate.
- Update PSSM_Semantics.xmi to reference the fUML 1.4 and PSCS 1.2 syntax and semantics models, as appropriate.
- Update PSSM_TestSuite.xmi to reference the fUML 1.4 library model.

Specification Document

2.3 Genericity of the Execution Model

- In the second paragraph, in the last sentence, replace "subclause 8.4.1.2.1 of [PSCS]" with "subclause 8.4.2.2 of [PSCS]".

[**Editorial Note.** Actually, it is the third paragraph.]

3 Normative References

Note: The following changes supersede the changes to this clause proposed in resolution PSSM_-5 -

https://issues.omg.org/browse/PSSM_-5 to issue PSSM_-4 - https://issues.omg.org/browse/PSSM_-4.

- In the reference to "[fUML]", replace "Version 1.2.1" with "Version 1.4".
- In the reference to "[PSCS]", replace "Version 1.0" with "Version 1.2".
- Remove the note "**Note.** The machine readable files for fUML and PSCS have been updated by this specification."

6.2 Changes to Adopted OMG Specifications

Note: The following change supersedes the changes to this subclause proposed in resolution PSSM_-5 - https://issues.omg.org/browse/PSSM_-5 to issue PSSM_-4 - https://issues.omg.org/browse/PSSM_-4.

- Remove this subclause and renumber subclause 6.3 to 6.2.

8.2 Values

In the first paragraph, at the end of the last sentence, change "(see [PSCS], 8.3.1.2.2)" to "(see [PSCS], 8.2.2.1)".

8.4 Common Behavior

In the first paragraph, in the third sentence, replace "(see [fUML], 8.4.3.2.7)" with "(see [fUML], 8.8.2.11)".

In the second bullet after the fourth paragraph, change "(see 8.4.3.1 in [fUML])" to "(see 8.8.1 in [fUML])".

8.5.2 State Machine Execution

Under the heading "StateMachineExecution", in the first paragraph, at the end of the first sentence, replace "(see [fUML], 8.4.2.1.1)" with "(see [fUML], 8.8.1)". In the fourth paragraph, second sentence, replace "(see [fUML], 8.4.3)" with "(see [fUML], 8.8.1)".

Under the heading "StateMachineEventAccepter", in the first paragraph, at the end of the second sentence, replace "(see [fUML], 8.4.3)" with "(see [fUML], 8.8.1)". In the second paragraph (the note), at the end of the first sentence, replace "(see [fUML], 8.6.4.2.1 and 8.6.4.2.2)" with "(see [fUML], 8.10.2.2 and 8.10.2.3)".

8.5.3 State Machine Semantic Visitors

Under the heading "StateMachineSemanticVisitor", in the third bullet after the third paragraph, replace "see [fUML], 8.2.2.2.6" with "see [fUML], 8.3.2.8".

8.5.6 "doActivity" Behavior Execution

Under the heading "DoActivityContextObject", in the first paragraph, at the end of the last sentence (before the bullets), replace "([fUML], Clause 8)" with "(see [fUML], 8.7.2.4)".

8.5.7.1 Basic Pseudostate Activations

Under the heading "JoinPseudostateActivations"/"Entry", at the end of the second numbered item, replace "see [fUML], 8.2.2.1" with "see [fUML], 8.3.1".

8.5.7.2 Connection Point Activations

Under the heading "EntryPointPseudostateActivation"/"Entry", at the end of item 3a, replace "see [fUML], 8.2.2.1" with "see [fUML], 8.3.1".

Under the heading "ExitPointPseudostateActivation"/"Enter", at the end of the first numbered item, replace "see [fUML], 8.2.2.1" with "see [fUML], 8.3.1".

8.5.7.3 Conditional Pseudostate Activations

Under the heading "ChoicePseudostateActivation"/"Entry", at the end the third numbered item, replace "see [fUML], 8.2.2.1" with "see [fUML], 8.3.1".

Under the heading "JunctionPseudostateActivation"/"Entry", at the end the second numbered item, replace "see [fUML], 8.2.2.1" with "see [fUML], 8.3.1".

8.5.9 Event Occurrences

In the first paragraph, in the third sentence, replace "(see [fUML], 8.4.3)" with "(see [fUML], 8.8.1)".

Remove the sections on the headings "CallEventExecution" and "CallEventExecution" (per PSSM_-5 - https://issues.omg.org/browse/PSSM_-5).

8.5.10.1 Event Triggered Execution

Under the heading "EventTriggeredExecution", in the first paragraph, in the last sentence (parenthetical), replace "see [fUML], 8.4.2" with "see [fUML], 8.8.1".

9.4.3 Transition

In the table row for "Transition 002" in the "Test(s)" column, replace "[fUML], 2.4" with "[fUML], 2.3".

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #2
Reporter:
Ed Seidewitz, Model Driven Solutions (ed-s@modeldriven.com)
Reported:
Fri, 26 Oct 2018 13:07 GMT on PSSM 1.0b1
Updated:
Fri, 9 Nov 2018 20:59 GMT
Discussion:
PSSM_-12 - https://issues.omg.org/browse/PSSM_-12

OMG Issue: PSSM_-23

Title:

Incorrect transition numbering in Deferred007_Test

Summary:

The classifier behavior of the test Deferred007 is specified as a state machine. Transitions in this state machine are numbered. However the numbering goes jumps from T3 to T6.

Source:

ad/2016-11-01 — Chapter/Section: 9.3.16.11

Resolution Summary:

Resolution of issue PSSM_-23 - https://issues.omg.org/browse/PSSM_-23

The resolution requires:

1. An update of the specification document. Indeed Figure 9.101, the generated trace, the note and the RTC steps overview need to updated.
2. An update of the test suite model.

Revised Text:

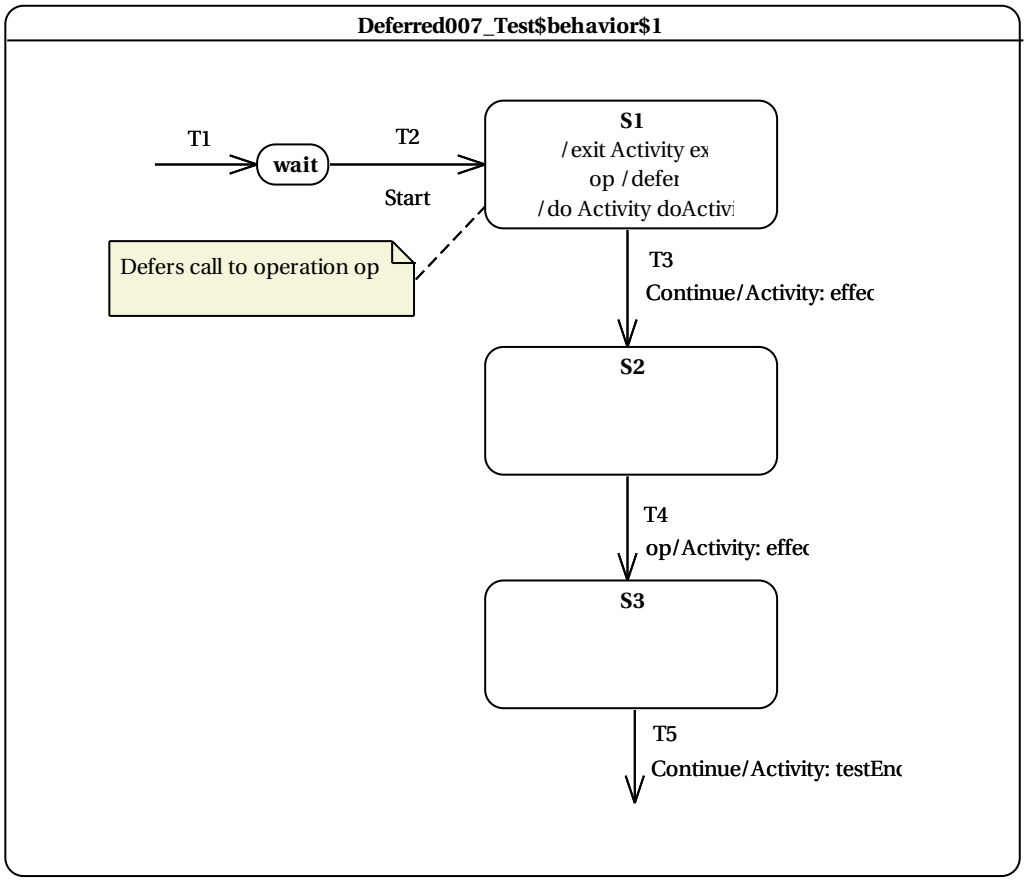
This update presumes that the resolution to PSSM_-2 is adopted

Specification Document

Update the specification document as follows.

9.3.16.11 Deferred 007

Replace the diagram in Figure 9.101 with:



Under the heading **Received event occurences**, replace

> Start – received when in configuration wait.
> op – received when in configuration S1.
> Continue – received when in configuration S1.
> Continue – received when in configuration S3.

with:

> Start – received when in configuration wait.
> Continue – received when in configuration S1.
> op – received when in configuration S1. Note: it can also be received when in configuration S2.
> Continue – received when in configuration S3.

Under the heading **Generated Trace**, update the generated trace to:

> S1(exit)::T3(effect)::T4(effect)[in=true][out=false]::[out=false]

Replace the note with:

> **Note**. The purpose of this test is to demonstrate that semantics of deferral also applies to call events. Consider the situation where the Start event is dispatched and accepted by the state machine. This implies T2 fires and S1 is entered. The doActivity of S1 is invoked and will be executed on its own thread of execution. The RTC step of the state machine ends, leaving the state machine in configuration S1. If the call to the operation op is received before the Continue signal event occurrence sent by the invoked doActivity, then the call event occurrence will be deferred by S1. This implies that the state machine remains in configuration S1 at the end of the RTC step. Next, the Continue signal event occurrence will be dispatched and accepted.This results in the call event occurrence for op being returned

to the event pool and transition T3 being fired. A completion event is generated for S2. The completion event is dispatched and lost due to lack of a completion transition outgoing S2. The next event to be dispatched is the call event occurrence. It is accepted by the state machine and triggers T4. A completion event is generated for S3 and lost when dispatched. When the Continue event occurrence sent by the tester is accepted by the state machine, T5 fires and the execution completes.

It is important to note that another execution is possible for this test case. Indeed, the Continue event occurrence sent by the doActivity may be added first to the event pool of the target. In this case, the call event occurrence has not been deferred yet. This means the state machine might be in configuration S2 at the point the call event occurrence gets accepted by the state machine. As a result, for this particular execution, it is not possible to assess the joint usage of call event and deferral semantics.

Under the heading **RTC Steps**

Replace the table with:

Step	Event pool	Deferred Events	State machine configuration	Fired transition(s)
1	[]	[]	[] - Initial RTC Step	[T1]
2	[Start, **CE(wait)**]	[]	[wait]	[]
3	[**Start**]	[]	[wait]	[T2]
4	[Call(op(true)), **CE(S1)**]	[]	[S1]	[]
5	[Continue, **Call(op(true))**]	[]	[S1]	[]
6	[**Continue**]	[Call(op(true))]	[S1]	[T3]
7	[Call(op(true)),**CE(S2)**]	[]	[S2]	[]
8	[**Call(op(true))**]	[]	[S2]	[T4]
9	[Continue, **CE(S3)**]	[]	[S3]	[]
10	[**Continue**]	[]	[S3]	[T5]

Test Suite Model

Update the model for the test Deferred 007, in package Deferred::007 in the PSSM test suite model, represented in the normative file PSSM_TestSuite.xmi, as follows.

Rename transition T6 as T4

Rename transition T7 as T5

Replace the Alf specification of the T6_effect behavior with:

T6 effect - Alf

```
activity T4_effect(in p: Boolean) : Boolean {
   this.trace("T4(effect)[in="+BooleanFunctions::ToString(p)+"][out="+BooleanFunctions::ToString(!p)+"]");
   return !p;
}
```

Replace the Alf specification of T4 effect behavior with:

T4 effect - Alf

```
activity effect(in p: Boolean) : Boolean {
 return 'T4_effect'(p);
}
```

Replace the Alf specification of the DeferredTests behavior with:

DeferredTests - Alf

```
namespace StateMachine_TestSuite::Deferred;

private import StateMachine_TestSuite::Util::Architecture::SemanticTestSuite;
private import StateMachine_TestSuite::Util::Architecture::SemanticTest;
private import StateMachine_TestSuite::Util::Architecture::A_tests_testSuite;

private import Deferred::'001'::'Deferred001_SemanticTest';
private import Deferred::'002'::'Deferred002_SemanticTest';
private import Deferred::'003'::'Deferred003_SemanticTest';
private import Deferred::'004-A'::'Deferred004_SemanticTest_A';
private import Deferred::'004-B'::'Deferred004_SemanticTest_B';
private import Deferred::'005'::'Deferred005_SemanticTest';
private import Deferred::'006-A'::'Deferred006_SemanticTest_A';
private import Deferred::'006-B'::'Deferred006_SemanticTest_B';
private import Deferred::'006-C'::'Deferred006_SemanticTest_C';
private import Deferred::'007'::'Deferred007_SemanticTest';

activity DeferredTests() {
 let suite : SemanticTestSuite = new SemanticTestSuite("Deferred");
 /*Deferred001*/
 d001 = new 'Deferred001_SemanticTest'();
 d001.name = "Deferred 001";
 d001.expectedTraces->add("S1(exit)::S2(entry)::T4(effect)::S3(entry)");
 suite.tests->add(d001);
 /*Deferred002 */
 d002 = new 'Deferred002_SemanticTest'();
 d002.name = "Deferred 002";
 d002.expectedTraces->add("S1(exit)::T4(effect)::S2(entry)::T6(effect)::S3(entry)");
 suite.tests->add(d002);
 /*Deferred003*/
 d003 = new 'Deferred003_SemanticTest'();
 d003.name = "Deferred 003";
 d003.expectedTraces->add("S1.1.1(exit)::T1.1.2(effect)::S1.1(exit)::T1.2(effect)::S1.2(exit)::T1.3(effect)");
 suite.tests->add(d003);
 /*Deferred004*/
 d004a = new 'Deferred004_SemanticTest_A'();
 d004a.name = "Deferred 004 A";
 d004a.expectedTraces->add("S1.1(exit)::T1.2(effect)::S2.1(exit)::T2.2(effect)::S1(exit)::T4(effect)");
 suite.tests->add(d004a);
 /*Deferred004-B*/
 d004b = new 'Deferred004_SemanticTest_B'();
 d004b.name = "Deferred 004 B";
 d004b.expectedTraces->add("S1.1.1(exit)::T1.1.2(effect)::S1.1(exit)::S2.1(exit)::T2.2(effect)::S1(exit)");
 suite.tests->add(d004b);
 /*Deferred005*/
 d005 = new 'Deferred005_SemanticTest'();
 d005.name = "Deferred 005";
 d005.expectedTraces->add("T3(effect)::S2(entry)::T4(effect)::S2(entry)::T5(effect)::S2(entry)");
 suite.tests->add(d005);
 /*Deferred006 A */
 d006 = new 'Deferred006_SemanticTest_A'();
 d006.name = "Deferred 006 A";
 d006.expectedTraces->add("S2(doActivity-AnotherSignal)");
 suite.tests->add(d006);
```

```
/*Deferred006 B */
d006b = new 'Deferred006_SemanticTest_B'();
d006b.name ="Deferred 006 B";
d006b.expectedTraces->add("");
d006b.expectedTraces->add("S2(doActivityPartI)::S2(doActivityPartII)");
suite.tests->add(d006b);
/*Deferred006 C */
d006c = new 'Deferred006_SemanticTest_C'();
d006c.name ="Deferred 006 C";
d006c.expectedTraces->add("S1.1(doActivity)::S1.2(doActivity)");
d006c.expectedTraces->add("S1.2(doActivity)::S1.1(doActivity)");
suite.tests->add(d006c);
/*Deferred007 */
d007 = new 'Deferred007_SemanticTest'();
d007.name ="Deferred 007";
d007.expectedTraces->add("S1(exit)::T3(effect)::T4(effect)[in=true][out=false]::[out=false]");
suite.tests->add(d007);
// Start test suite
suite.Start();
}
```

Extent Of Change:
Significant
Disposition:
Resolved - Approved on Ballot #3
Reporter:
Jeremie Tatibouet, Commissariat a l Energie Atomique-CEA (jeremie.tatibouet@cea.fr)
Reported:
Wed, 7 Nov 2018 15:55 GMT on PSSM 1.0a1
Updated:
Mon, 12 Nov 2018 00:16 GMT
Discussion:
PSSM_-23 - https://issues.omg.org/browse/PSSM_-23
Attachments:

- Deferred007_Test.svg - ptc-18-11-07/PSSM_-23/Deferred007_Test.svg 23 kB (image/svg+xml)

Disposition: Deferred

OMG Issue: PSSM_-1

Title:

Tests that send multiple signals are not correct

Summary:

Any test in the PSSM test suite with a test driver that sends multiple signals to the model being tested may not currently be properly allowing all possible execution traces. This is because itt cannot, in general, be presumed that event occurrences are received in the order they are sent, even if they are all sent from the same thread. This was always true in fUML (per the statement of "the semantics of inter-object communications mechanisms" in subclause 2.4 of the spec), but it is completely, formally clear in fUML 1.3, in which EventOccurrence is an active class, such that all event occurrences are sent concurrently with each other.

For example, consider test Transition 007 (described in subclause 9.3.3.2 of the PSSM beta spec). The tester behavior for this test sequentially sends three signal instances: AnotherSignal, Continue and Continue again. However, while these signals are sent sequentially, there is no guarantee they will be received by the tested state machine in the same order. For example, one of the Continue signal instances could be received before the AnotherSignal instance, which would cause the state machine (as shown in Fig. 9.12) to take transition T3 to S2 and never get to S3.

Rather than try to capture all the possible traces that should be allowed by the such tests a s currently modeled, it would be better to modify the tests so that they should only produce the trace that is currently suspected. This can be done by having the state machine under test send signals back to the tester, in order to coordinate the sending of sequential signals. For example, in the case of Transition 007, the state machine could send signals back to the tester as part of the doTraversial behaviors for transitions T1 and T2. The test behavior would then have to include accept event actions in order to wait between the send signal actions. (Of course, to allow the test to send signals back to the tester, either the Tester/Target association in the test architecture would need to be made bidirectional, or some signaling mechanism would need to be provided through the SemanticTest class.)

Source:

ptc/17-04-04 — Chapter/Section: 9

Resolution Summary:

Defer

While the FTF agrees that this is an issue that should be resolve, it is deferred to the first RTF due to lack of time.

Disposition:

Deferred - Approved on Ballot #2

Reporter:

Ed Seidewitz, Model Driven Solutions (ed-s@modeldriven.com)

Reported:

Tue, 5 Dec 2017 23:11 GMT on PSSM 1.0b1

Updated:

Fri, 9 Nov 2018 00:05 GMT

Discussion:

PSSM_-1 - https://issues.omg.org/browse/PSSM_-1

OMG Issue: PSSM_-3

Title:

PSSM implementation shall conform to bUML

Summary:

The behavioral part of the PSSM semantic model is specified using Java syntax. The subset of Java that is used does not always conform to the mapping rules defined in Annex A of fUML between Java and Activities.

Examples:

1. Usage of index starting from 0 instead of 1 in StateActivation::hasCompleted operation.
2. Constructor call with arguments in StateMachineEventAccepter::accept operation.
3. Usage of an iterative for loop instead a parallel for loop in StateActivation::enterRegion operation.

Source:
ad/2016-11-01

Resolution Summary:

Defer

The FTF agrees that this issue should be resolved, but it is deferred to the first RTF due to lack of time. Further, while the formal structure of fUML base semantics requires the use of bUML, the standard is still clear as it is on the functionality being specified, even if not fully conformant to bUML.

Disposition:
Deferred - Approved on Ballot #2

Reporter:
Jeremie Tatibouet, Commissariat a l Energie Atomique-CEA (jeremie.tatibouet@cea.fr)

Reported:
Thu, 13 Apr 2017 13:02 GMT on PSSM 1.0b1

Updated:
Fri, 9 Nov 2018 00:05 GMT

Discussion:
PSSM_-3 - https://issues.omg.org/browse/PSSM_-3

OMG Issue: PSSM_-6

Title:

Notation for entry, do and exit behaviors is wrong

Summary:

The correct notation is for example:

entry/Activity entry

All the diagrams in PSSM show instead

/entry Activity entry

This is confusing and should get changed.
Also the text in all examples of the UML specification is left justified. It is not mentioned as a requirement, but I think most tools follow this convention. In the PSSM specification the text is centered. I suggest to change it to left justified.
The effect of Transitions is notated with a colon:

/Activity: effect.

I think that should also be consistent. Either remove the colon, or use it with state behaviors as well.
As an additional suggestion: In most cases it is not relevant for the test case, that an activity is called. The string "Activity" could be left out to keep the diagram less cluttered.

Source:
ptc/2017-04-01 — Chapter/Section: all — Page Number/s: all

Resolution Summary:

Defer

This is the result of a bug in the Papyrus tooling used to generate the diagrams. Since the notation currently in the specification is still clear, if not entirely correct, the FTF proposes to defer the resolution of this issue until a fix is available for Papyrus.

Disposition:
Deferred - Approved on Ballot #2

Reporter:
Axel Scheithauer, oose Innovative Informatik eG (axel.scheithauer@oose.de)

Reported:
Tue, 3 Apr 2018 15:31 GMT on PSSM 1.0b1

Updated:
Fri, 9 Nov 2018 00:05 GMT

Discussion:
PSSM_-6 - https://issues.omg.org/browse/PSSM_-6

Disposition: Transfered

No issues in this report.

Disposition: Closed; No Change

No issues in this report.

Disposition: Closed; Out Of Scope

No issues in this report.

Disposition: Duplicate or Merged

OMG Issue: PSSM_-11

Title:

"Join 003" has (invalid) triggers on transitions entering join

Summary:

According to the UML 2.5.1 spec, there's a constraint `join_segment_guards` which prohibits transitions whose target is a join pseudostate from having guards or triggers (the constraint's scope is larger than its name would imply). But the sample state machine "Join 003" has triggers on the two transitions entering the join.

Essentially, as I read the UML spec, transitions leading to joins can **only** be triggered by completion events.

I don't think that restriction is necessary, and can be trivially worked around (make a composite state with a triggered transition entering the final state). But this state machine does appear to violate that constraint.

Source:

ptc/17-04-04 — Chapter/Section: 9.3.12.4 Join 003 — Page Number/s: 179

Resolution Summary:

Merge with PSSM_-8 - https://issues.omg.org/browse/PSSM_-8

This issue relates to the same element of the same test case, Join 003, as issue PSSM_-8 - https://issues.omg.org/browse/PSSM_-8. The two issues can therefore be resolved together.

Disposition:

See Issue PSSM_-8 - https://issues.omg.org/browse/PSSM_-8 for disposition - Approved on Ballot #2

Reporter:

Daniel Yankowsky, AGI (dyankowsky@agi.com)

Reported:

Thu, 20 Sep 2018 18:16 GMT on PSSM 1.0b1

Updated:

Fri, 9 Nov 2018 00:05 GMT

Discussion:

PSSM_-11 - https://issues.omg.org/browse/PSSM_-11